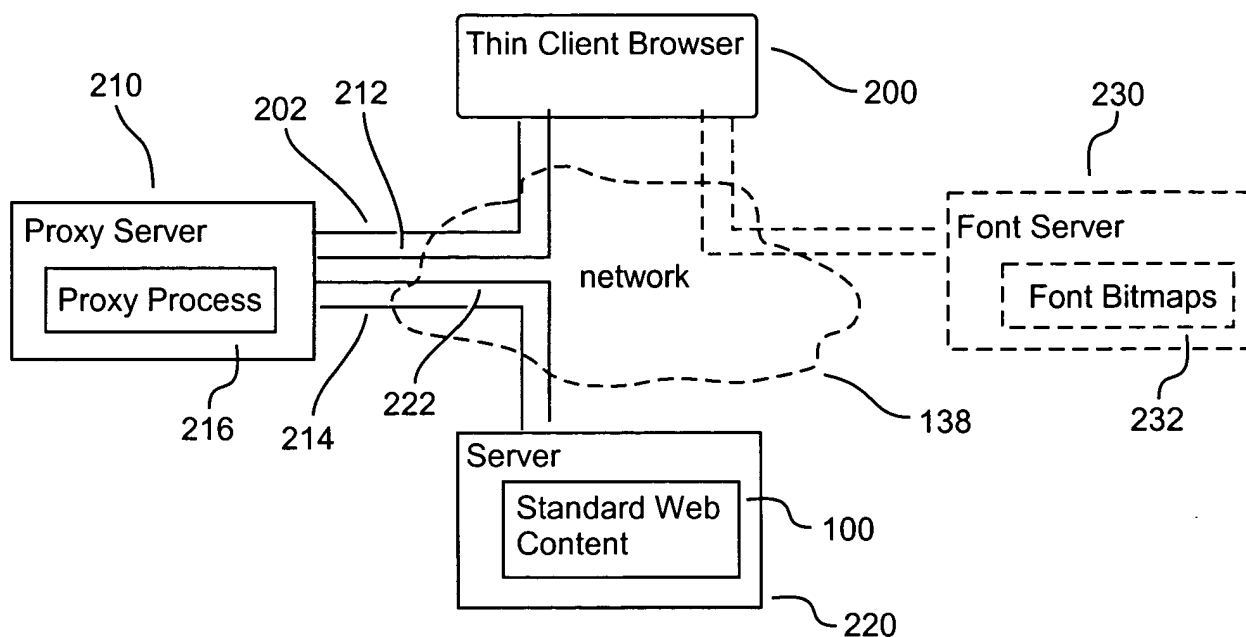
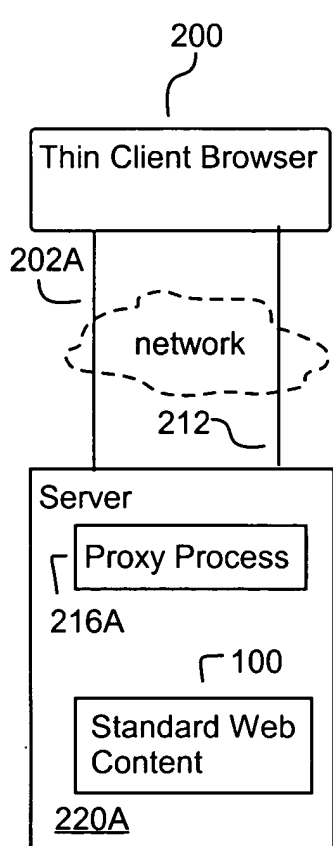


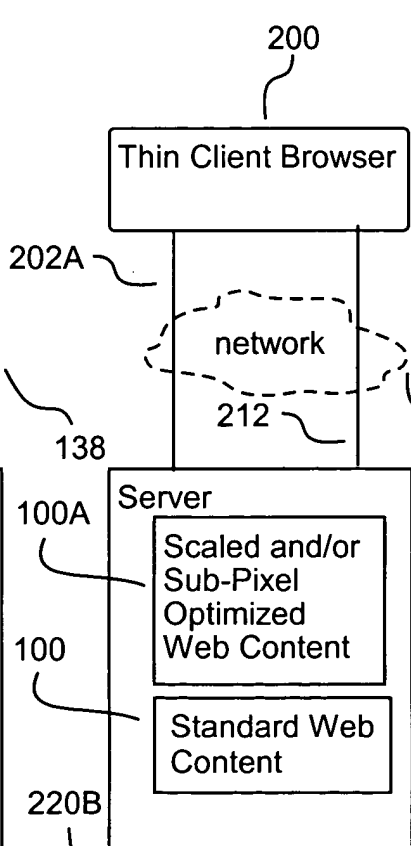
FIG. 1



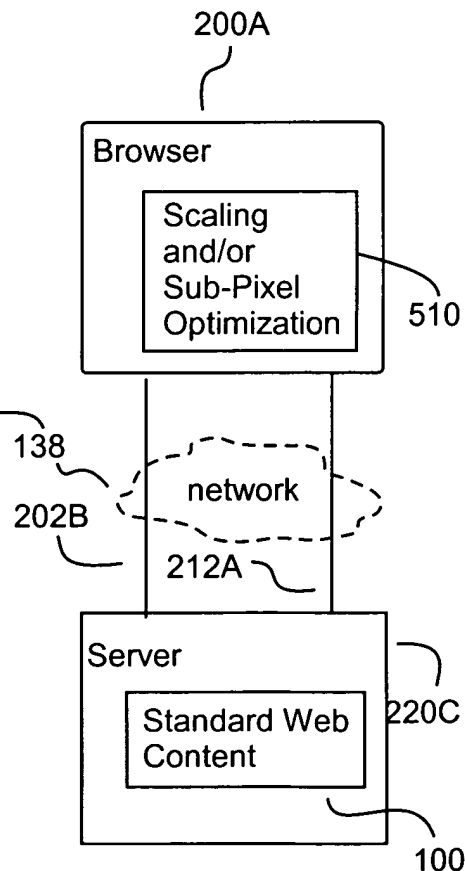
**FIG. 2**



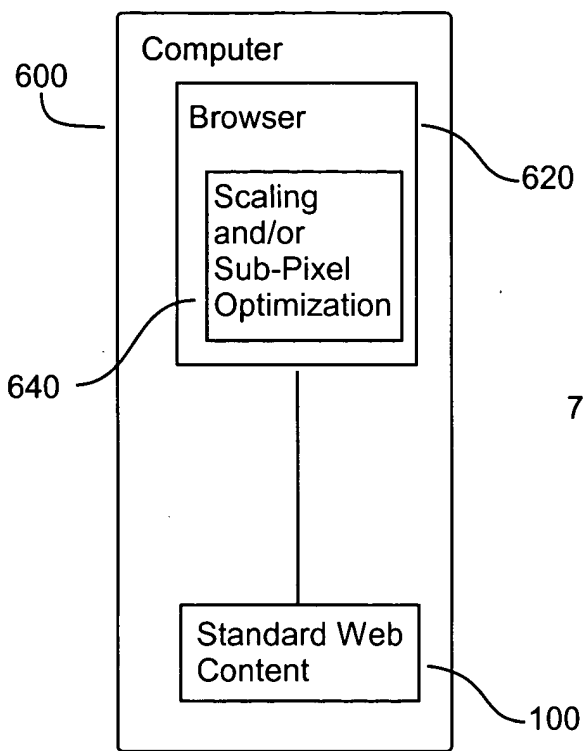
**FIG. 3**



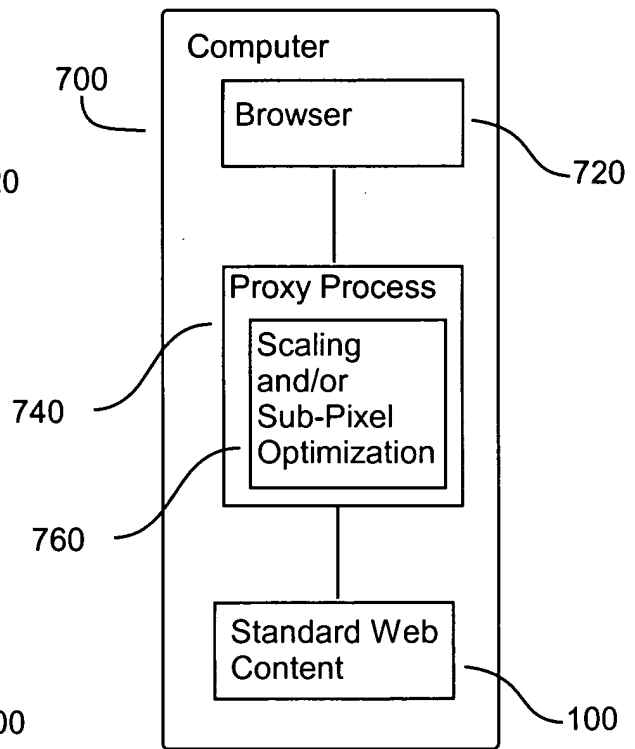
**FIG. 4**



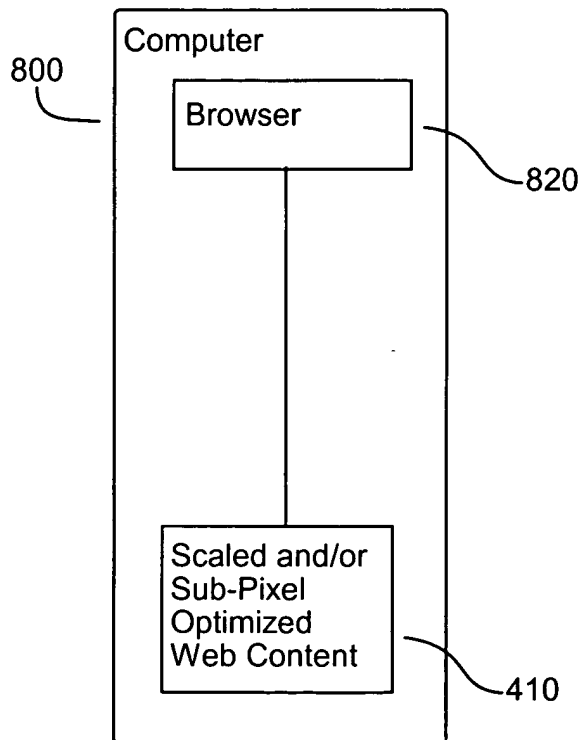
**FIG. 5**



**FIG. 6**



**FIG. 7**



**FIG. 8**

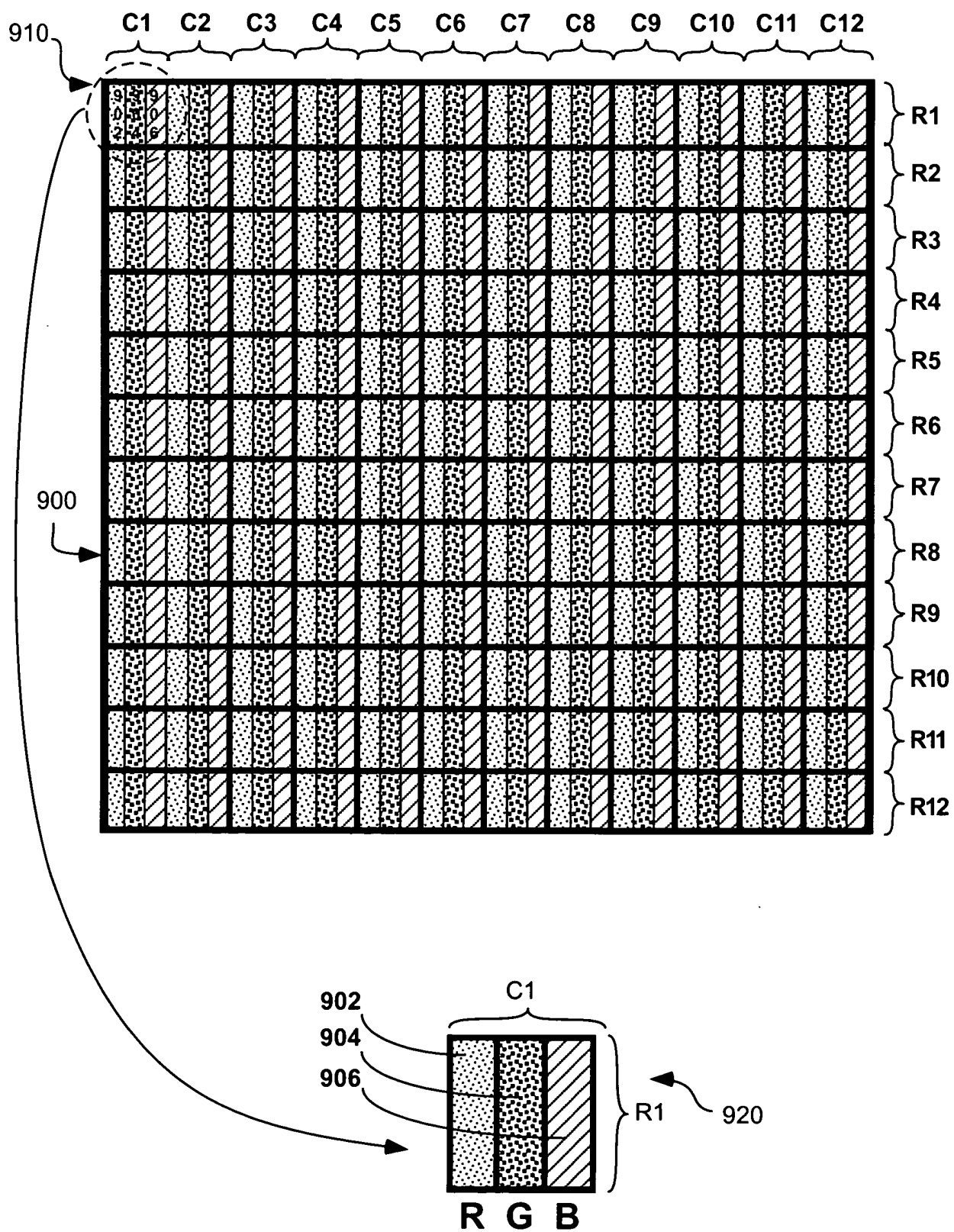


FIG. 9

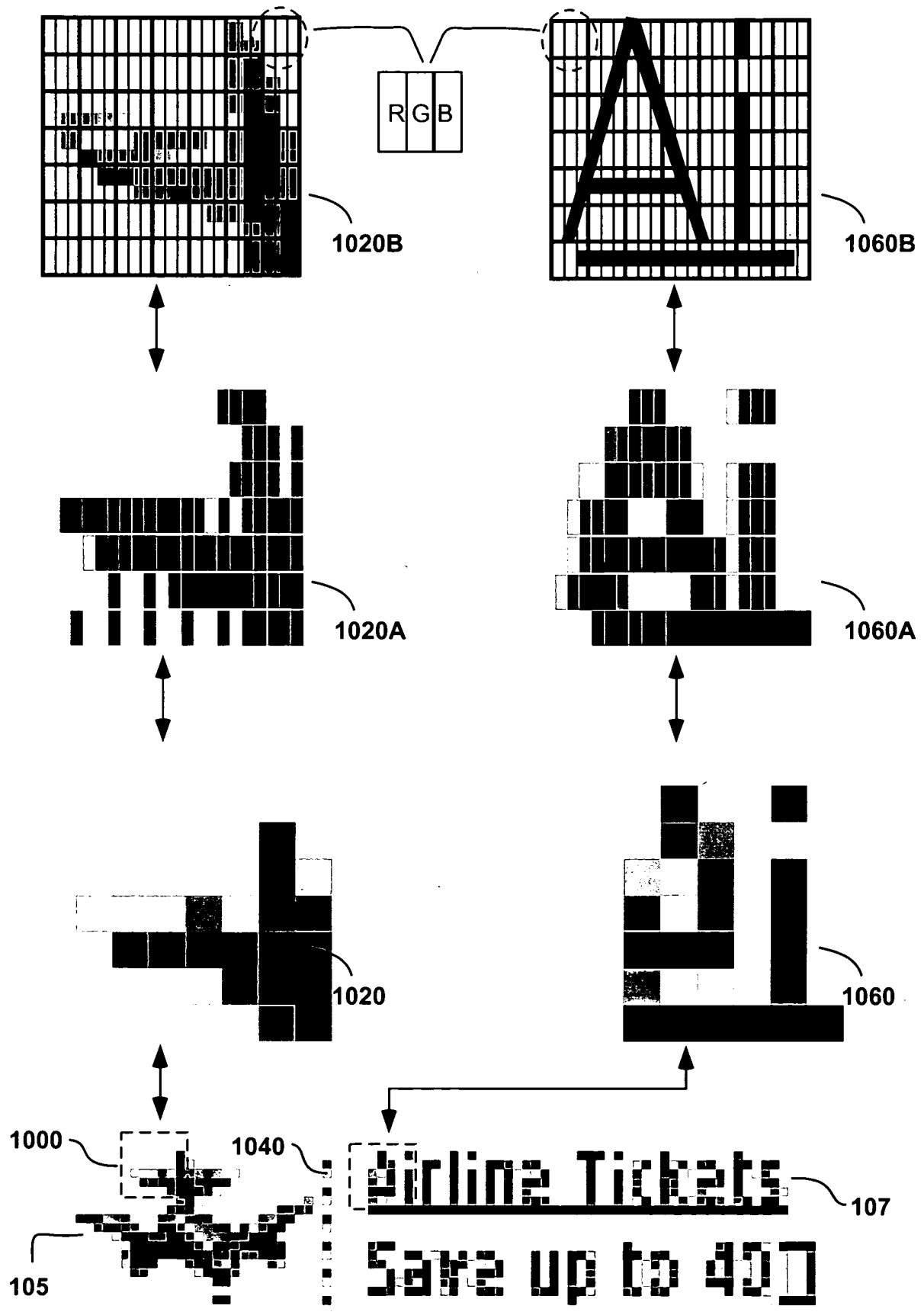


FIG. 10

Detail From Standard 640 x 480 layout

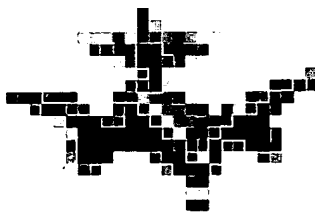


## Airline Tickets

Save up to 40% or  
more in 15 minutes or  
less!

1100

Detail From SubPixel Optimized 320 x 240 layout (shown on non-subpixel display)

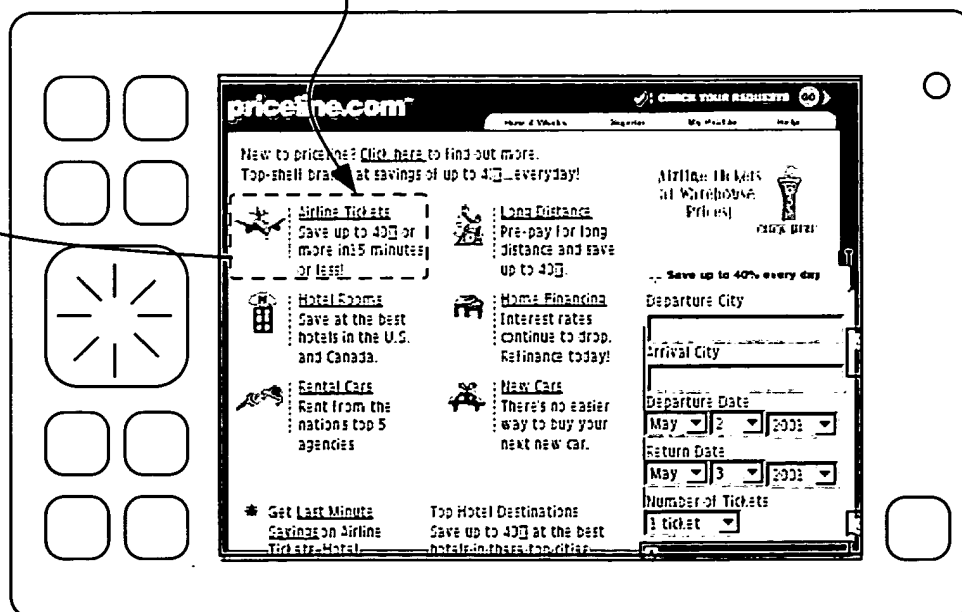


## Airline Tickets

Save up to 40% or  
more in 15 minutes  
or less!

1120

1130

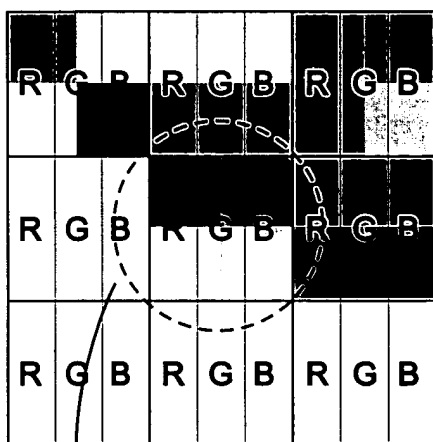


200B

FIG. 11

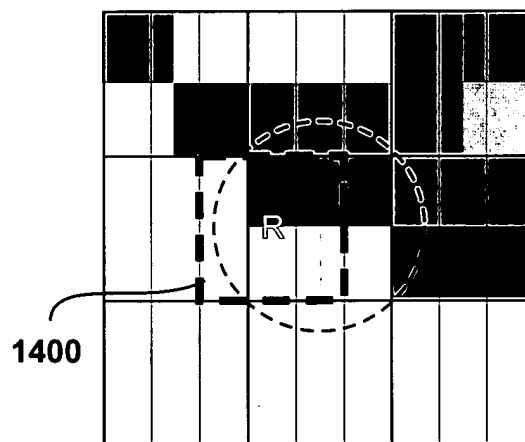


FIG. 12



1300

FIG. 13



1400

FIG. 14

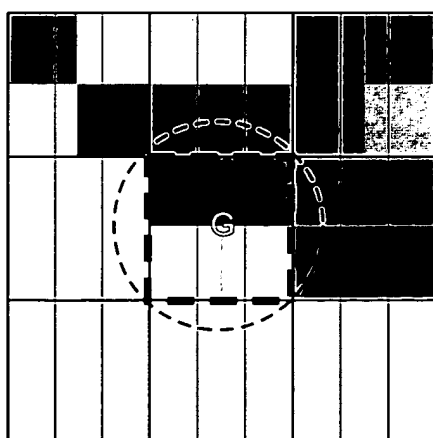


FIG. 15

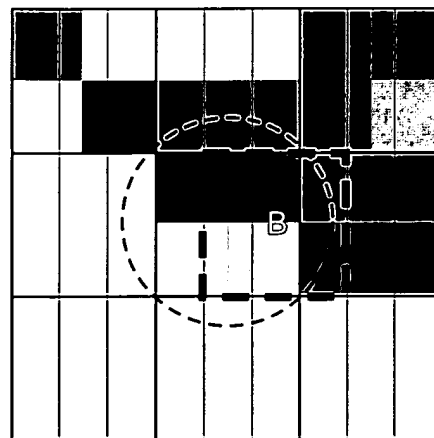


FIG. 16

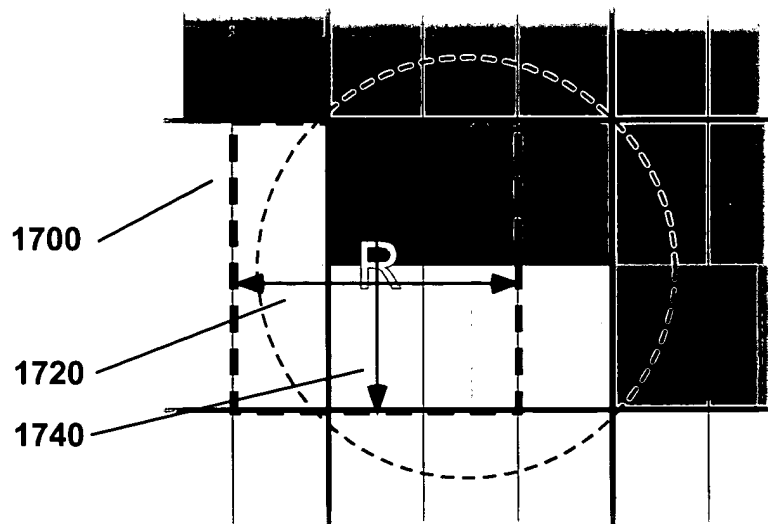


FIG. 17

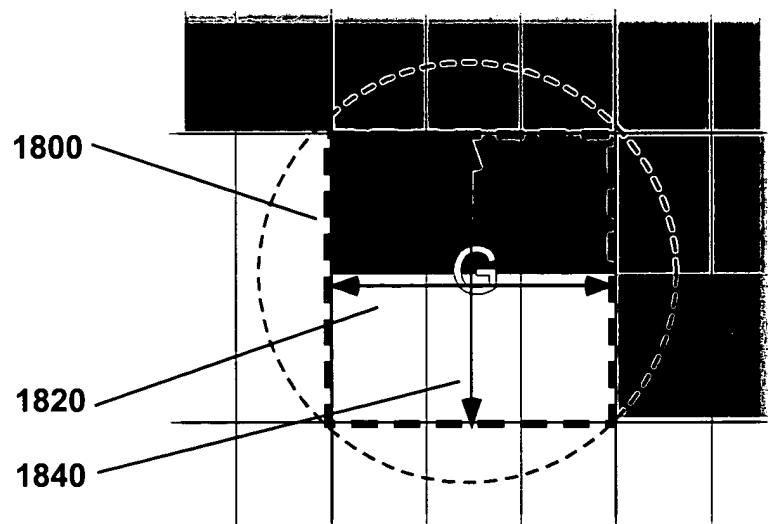


FIG. 18

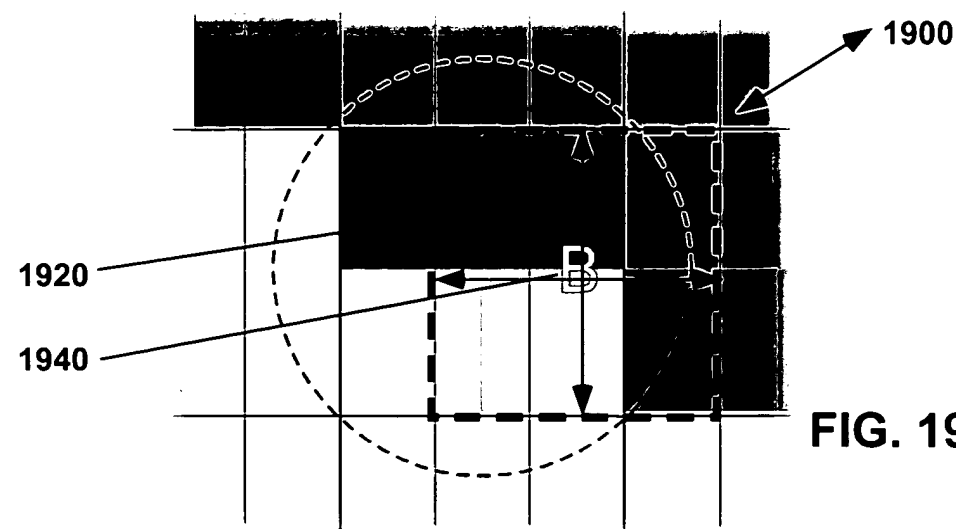


FIG. 19



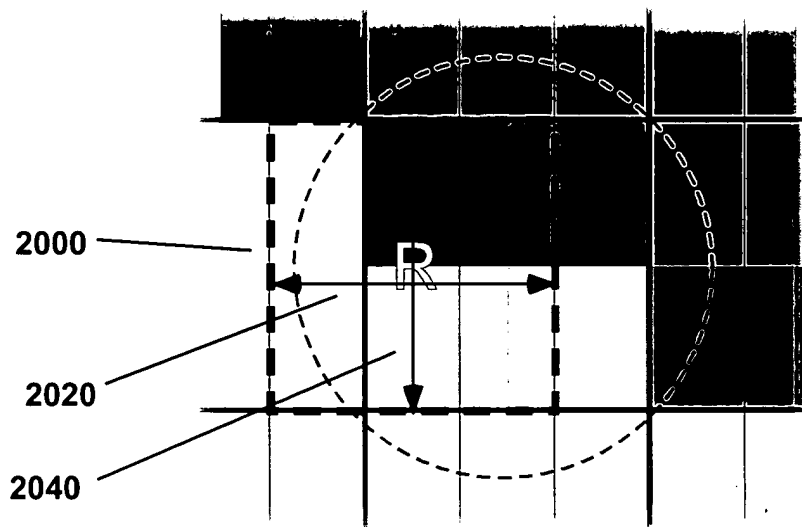


FIG. 20

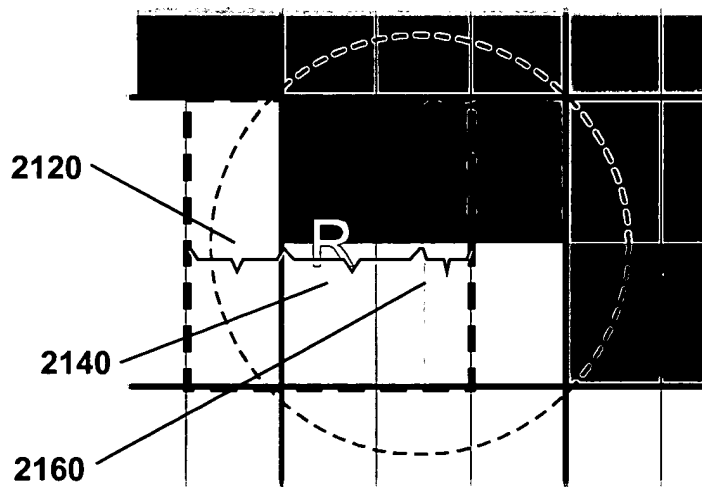


FIG. 21

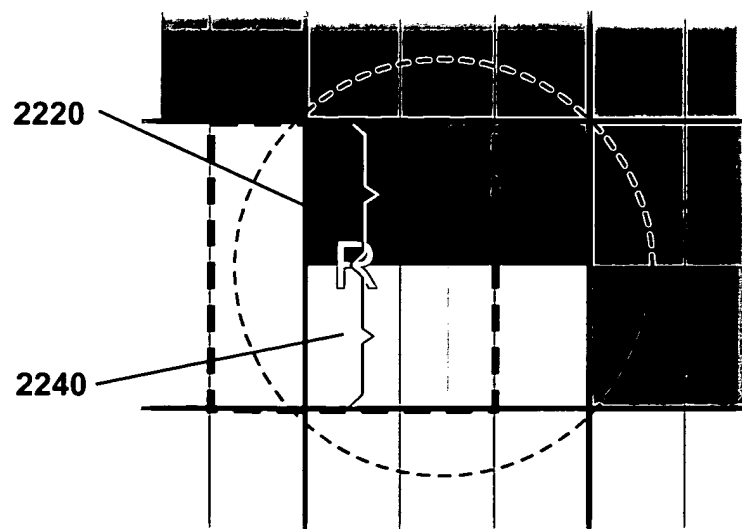
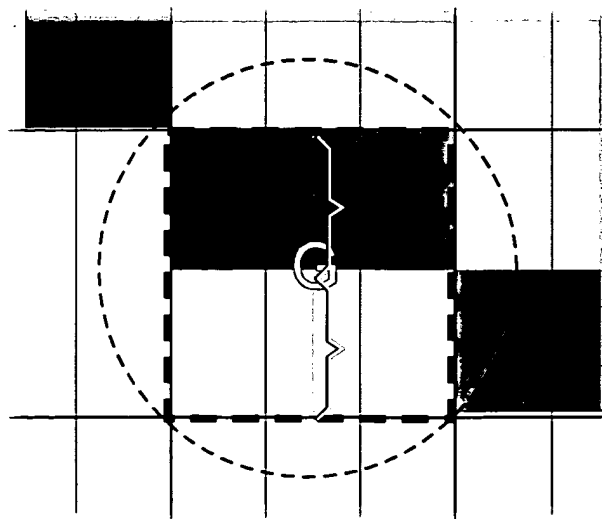
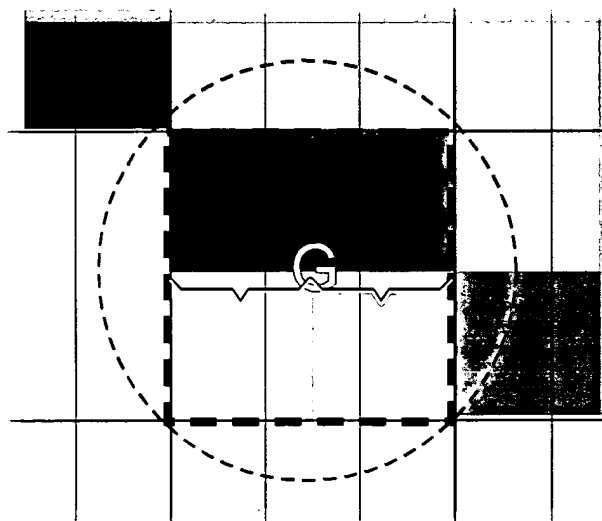
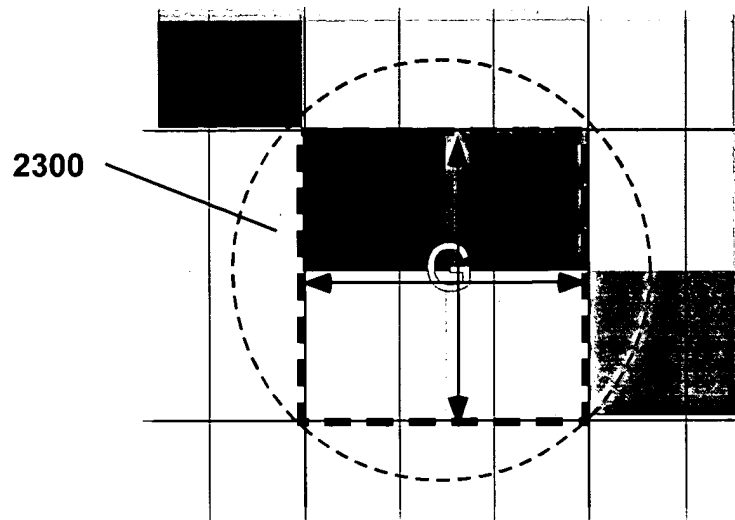
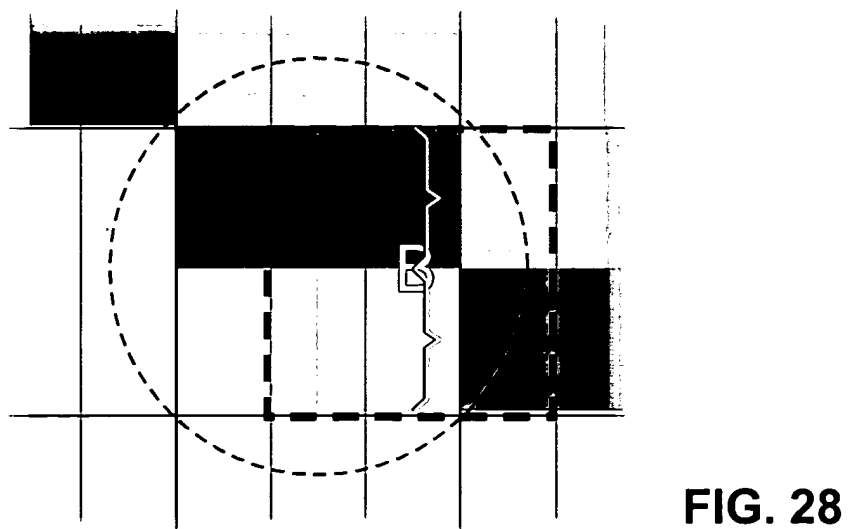
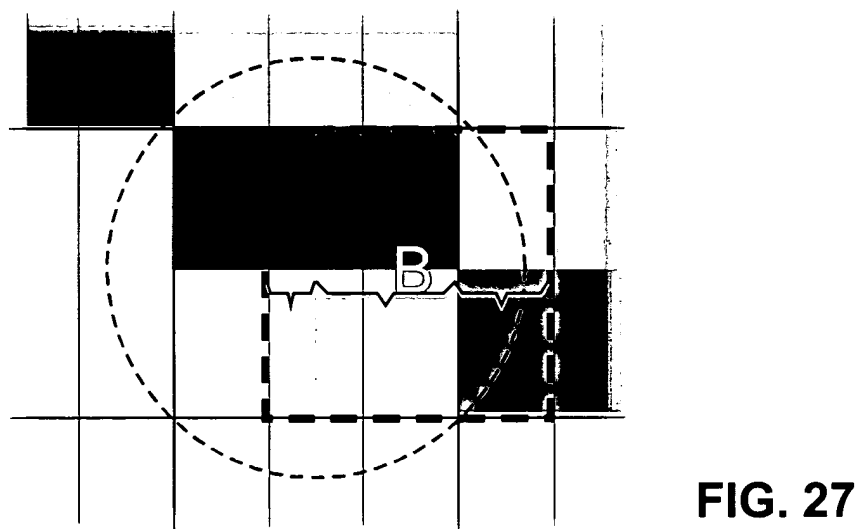
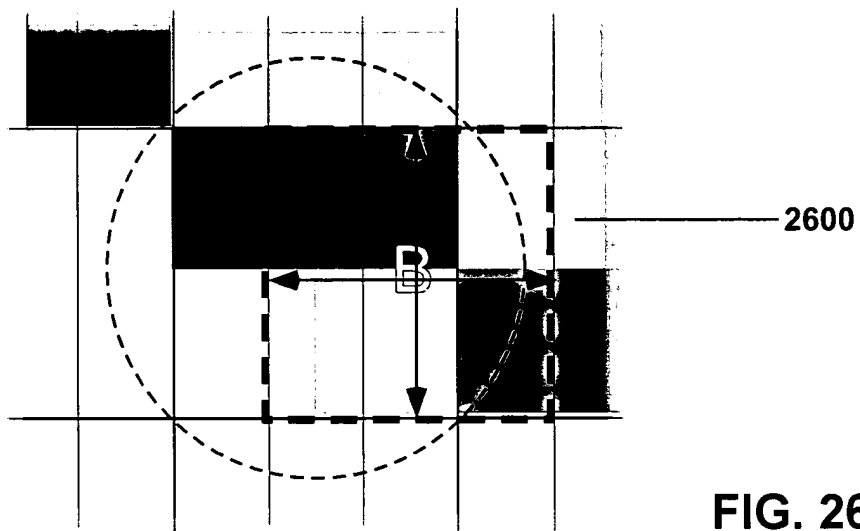


FIG. 22





- subpixel optimization routine (using line coverage)~2900
  - for each row in output image~2901
    - for each pixel in row~2902
      - for each subpixel in pixel~2904
        - for each of its scan lines~2906
          - calculate each intersection between that scan line and a pixel boundary~2908
          - for each portion of a scan line which occurs between two scan line ends, a scan line end and a pixel boundary, or two pixel boundaries~2910
            - add to a coverage value associated with the subpixel the multiple of the percent of that scan line covered by the portion times the component color value of the pixel covering that portion corresponding to the color of the current subpixel, all divided by the number of the subpixel's scan lines~2912
    - Set the pixel's color value equal to a color having an compound RGB value with red, green, and blue values equal to the subpixel luminosity value of the pixel's red, green, and blue subpixels, respectively.~2914

**FIG. 29**

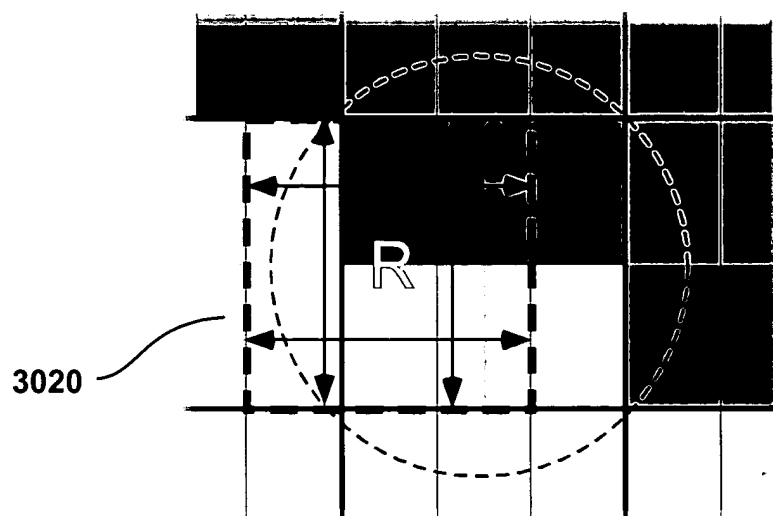


FIG. 30

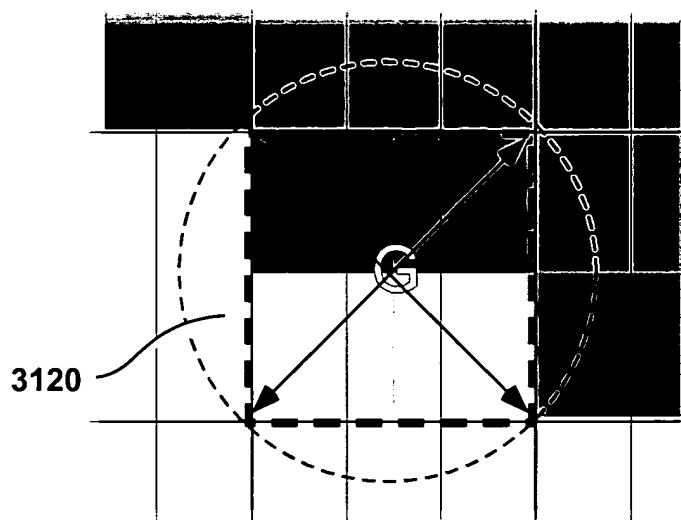


FIG. 31

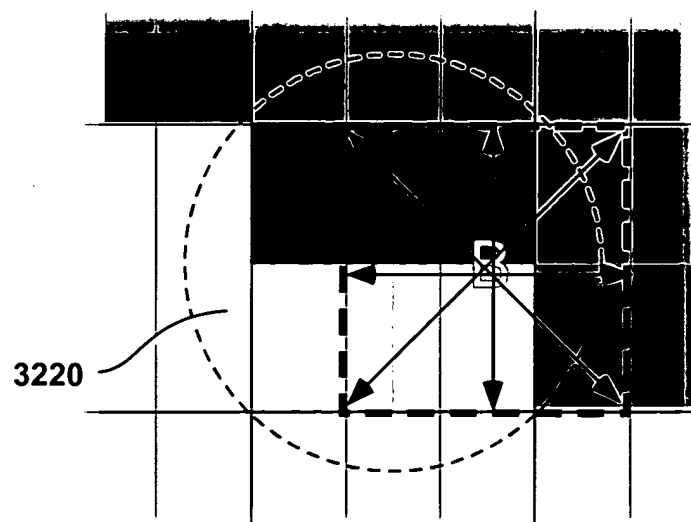
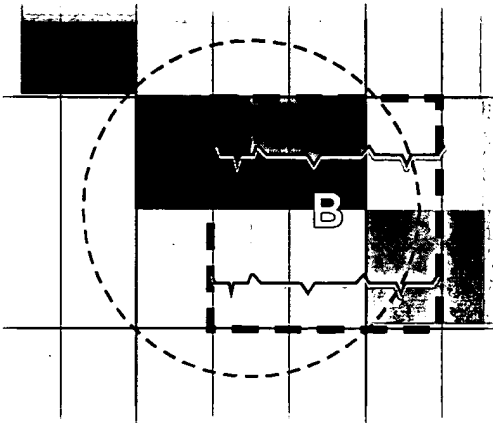
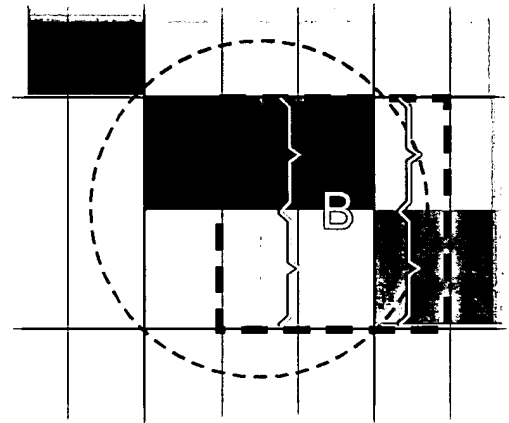


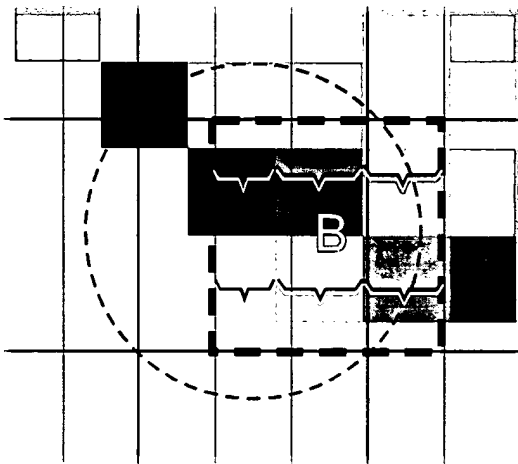
FIG. 32



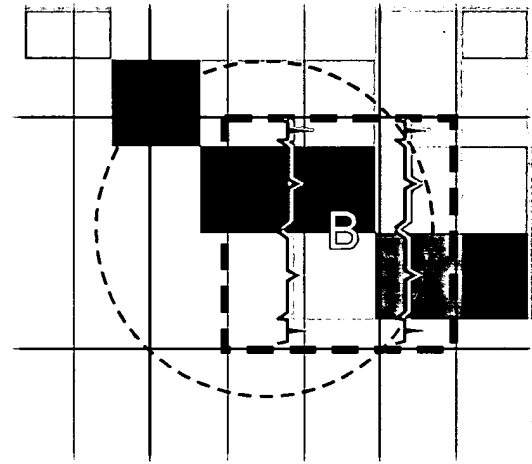
**FIG. 33**



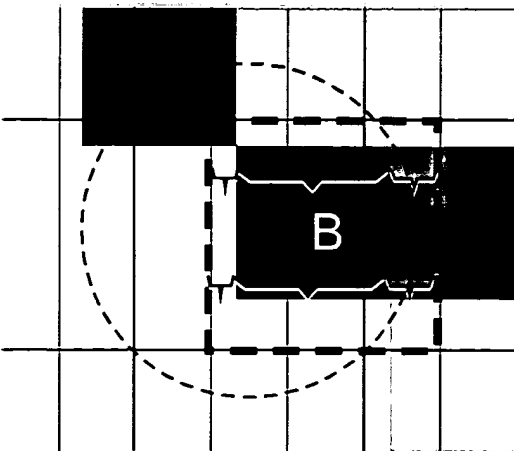
**FIG. 34**



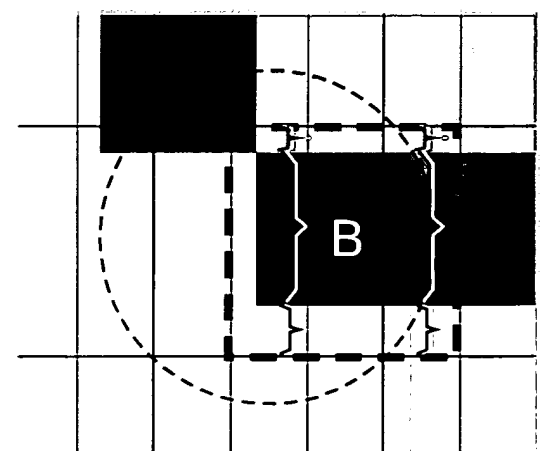
**FIG. 35**



**FIG. 36**



**FIG. 37**



**FIG. 38**

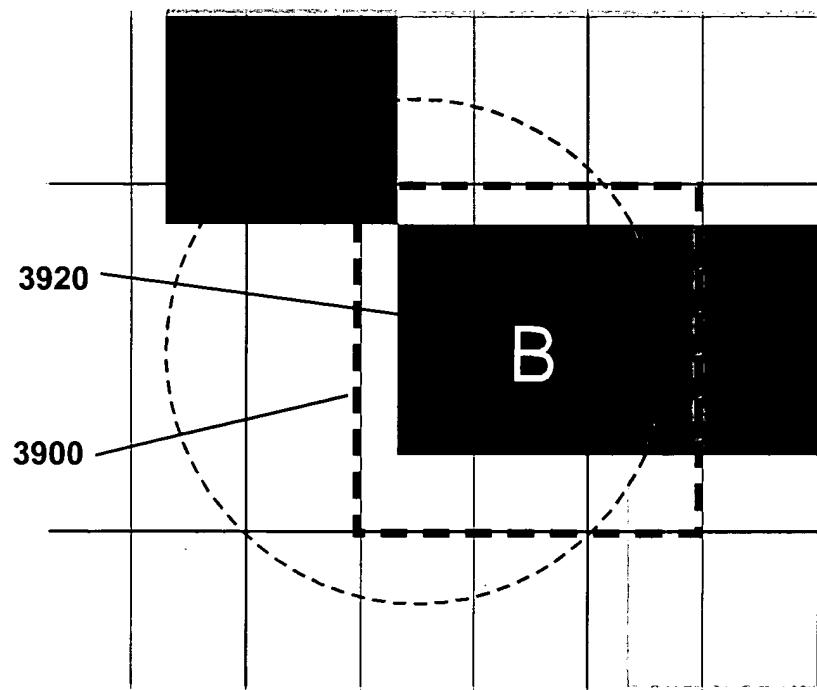


FIG. 39

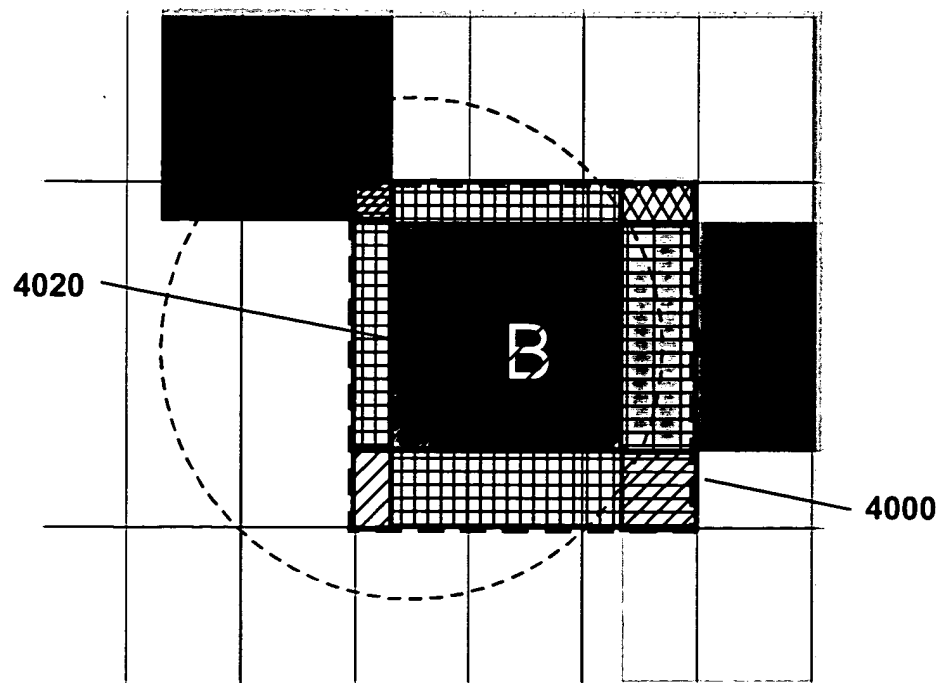


FIG. 40

- subpixel optimization routine (using area coverage)~4100
  - for each pixel row in output image~4102
    - for each pixel in row~4104
      - for each subpixel in a pixel~4106
        - determine which pixels of source image are in a source image window associated with the subpixel~4108
        - for each such included source image pixel~4110
          - calculate the percent of the subpixel's source image window's area covered by the pixel's area~4112
          - add to a luminosity value being calculated for the subpixel the multiple of the percentage of the subpixel's source image window area covered by the pixel times the pixel's color component value corresponding to the color of the current subpixel~4114
  - Set the pixel's color value equal to a color having an compound RGB value with red, green, and blue values equal to the subpixel luminosity value calculated for the pixel's red, green, and blue subpixels, respectively.~4116

**FIG. 41**



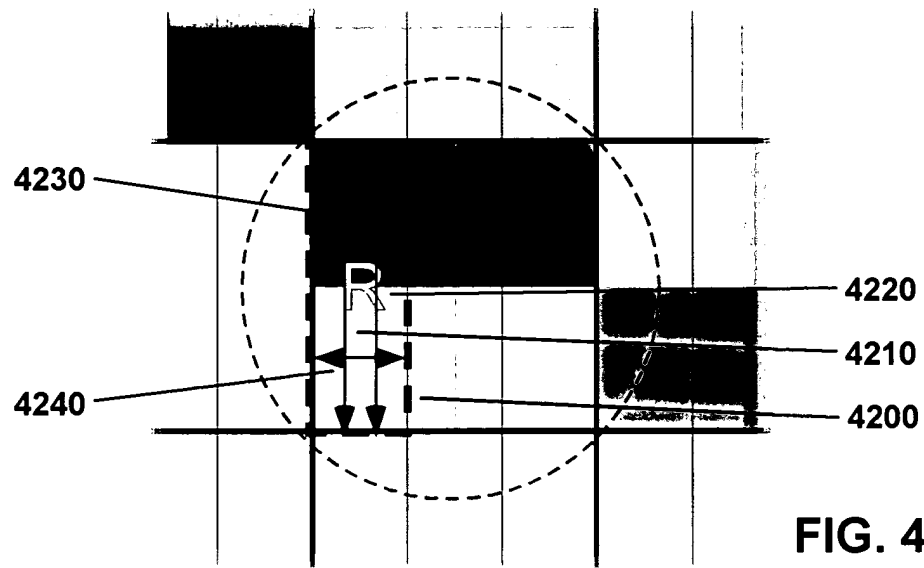


FIG. 42

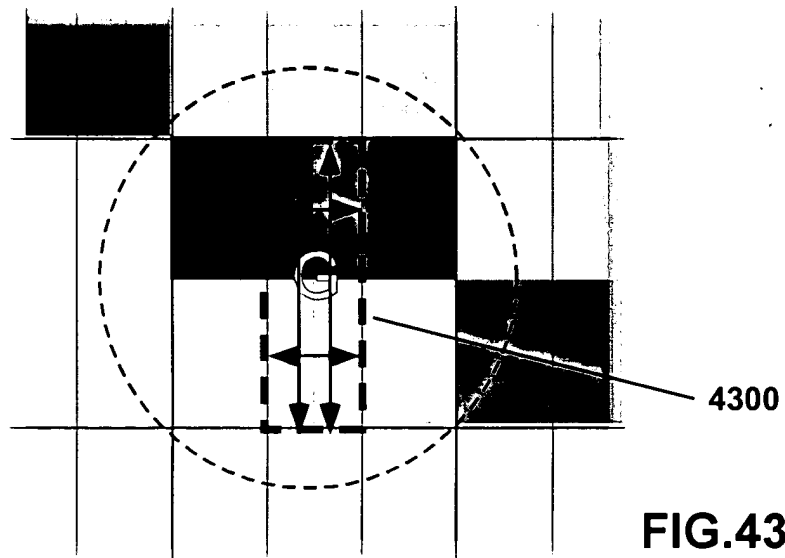


FIG. 43

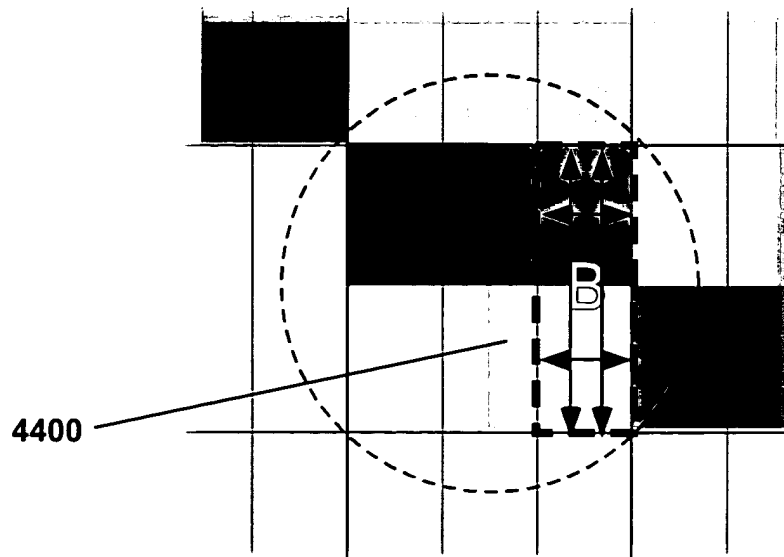
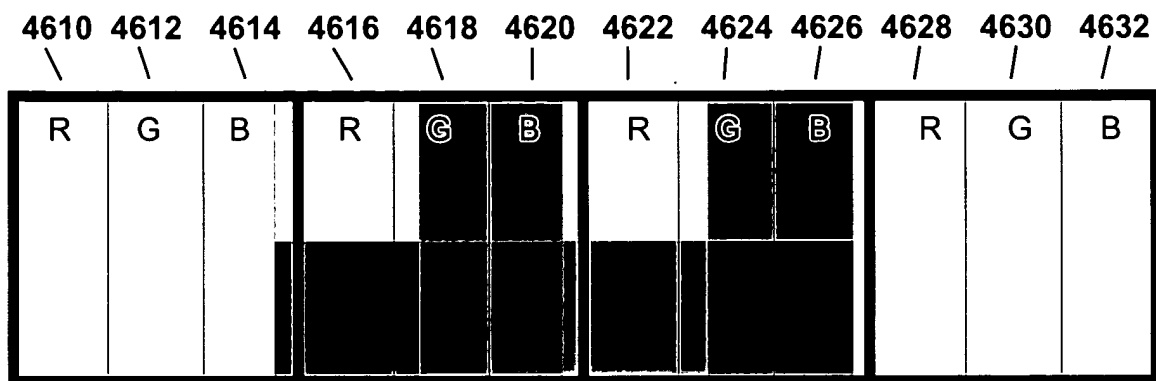
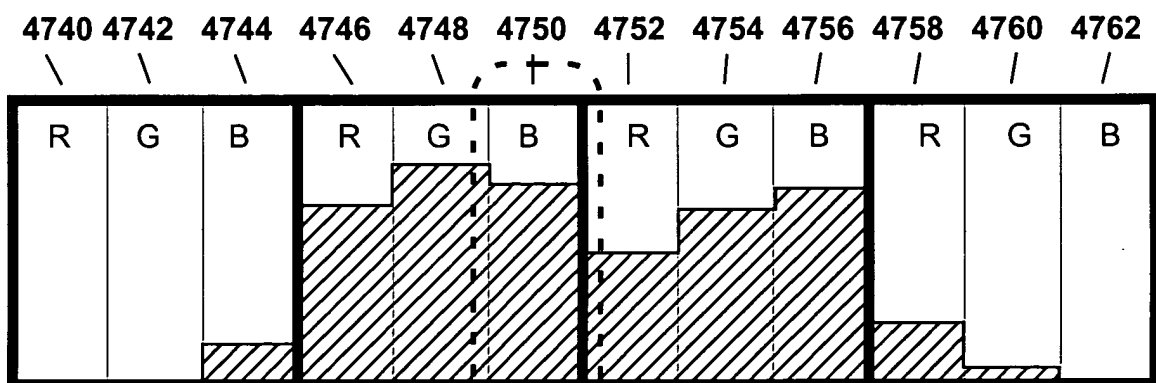


FIG. 44

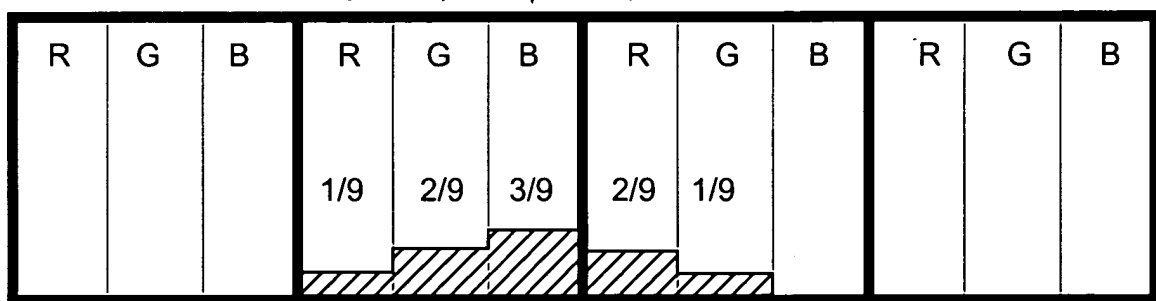


4600

FIG. 45



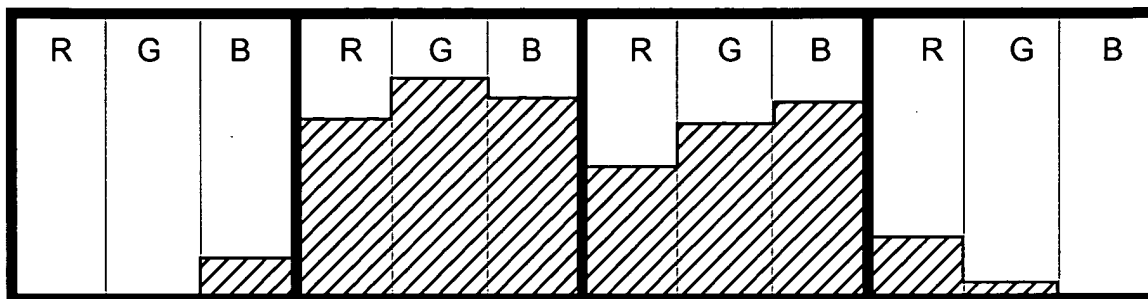
4700



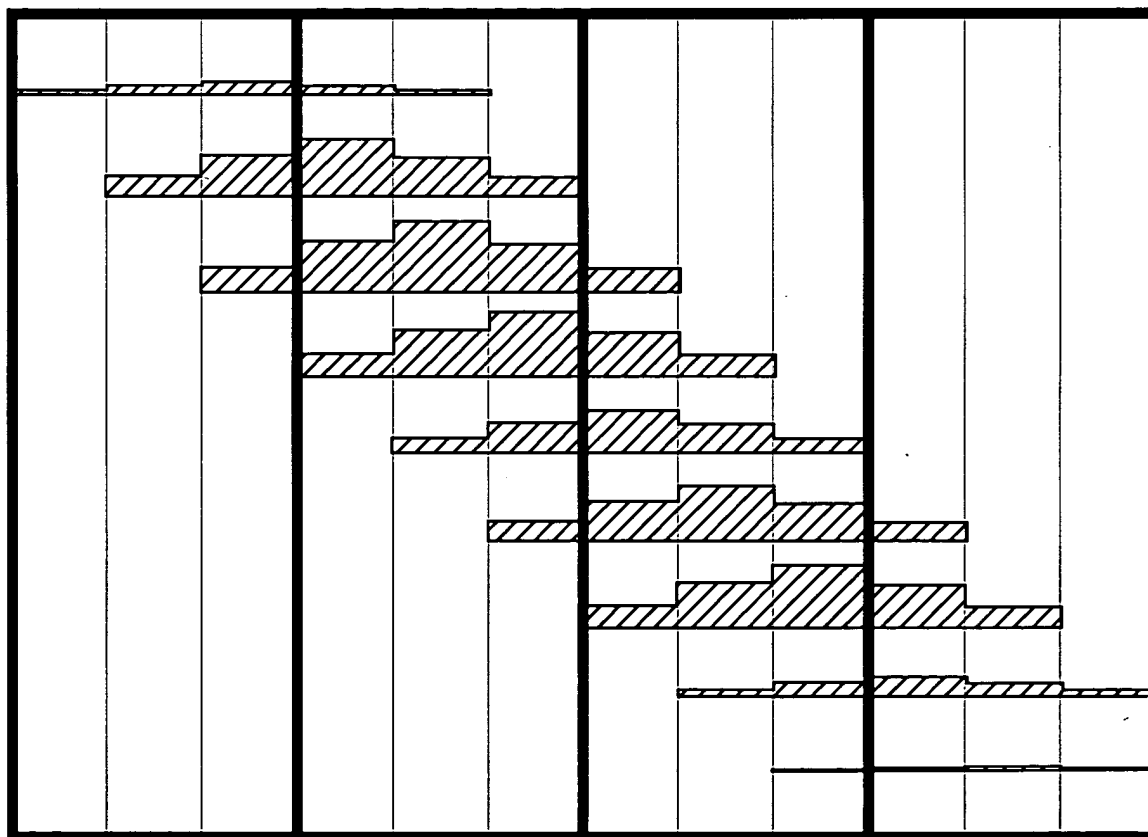
4702

FIG. 46

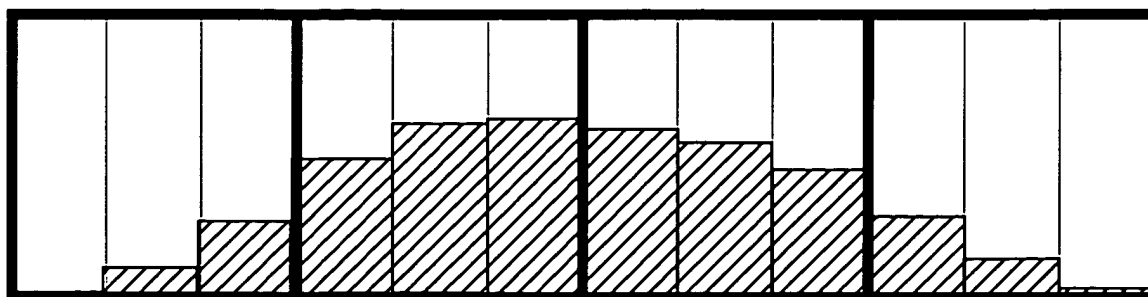
4740 4742 4744 4746 4748 4750 4752 4754 4756 4758 4760 4762  
 \ \ \ \ \ | | | / / /



4800 ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ 4802



↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓



4804

FIG. 47

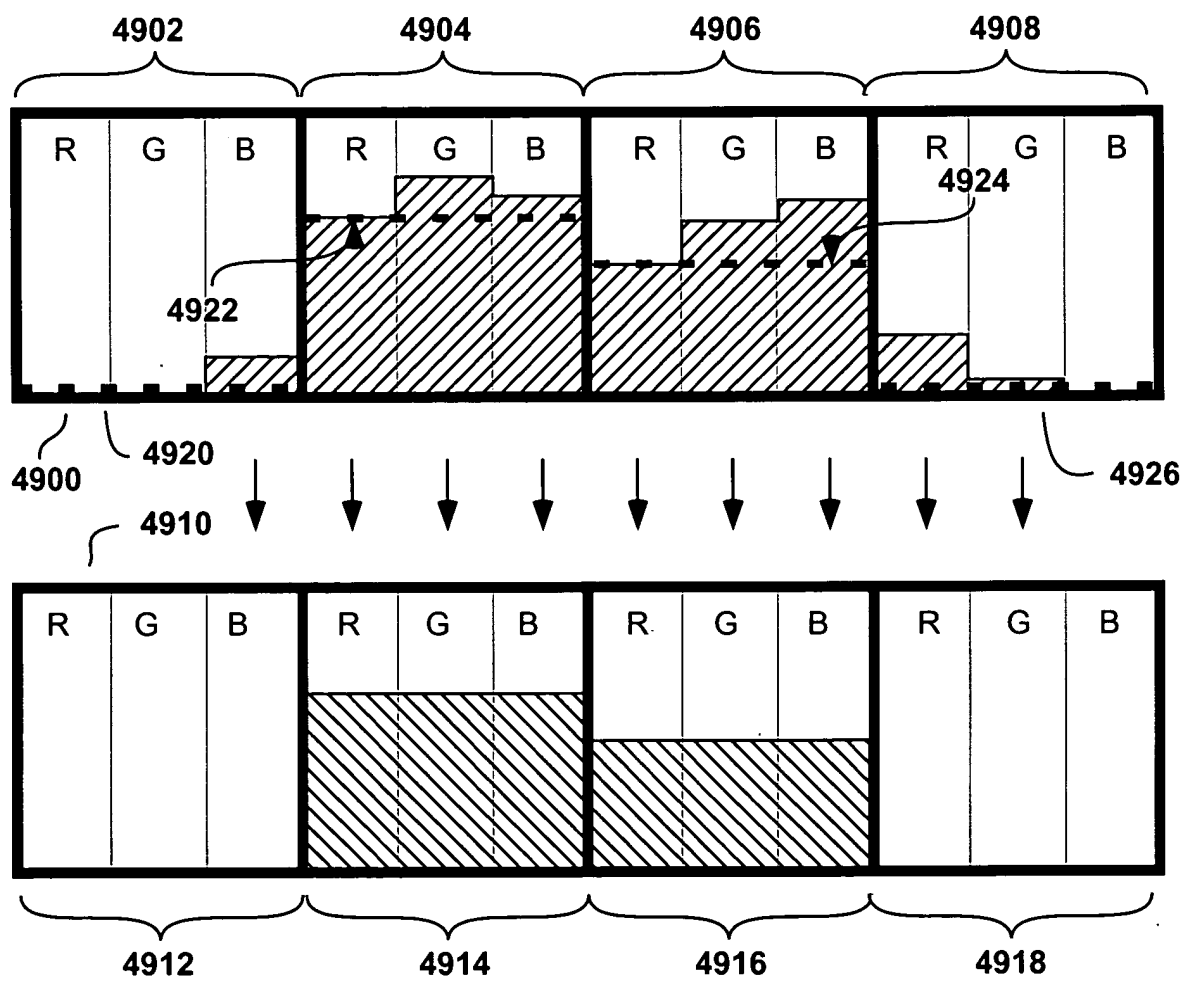
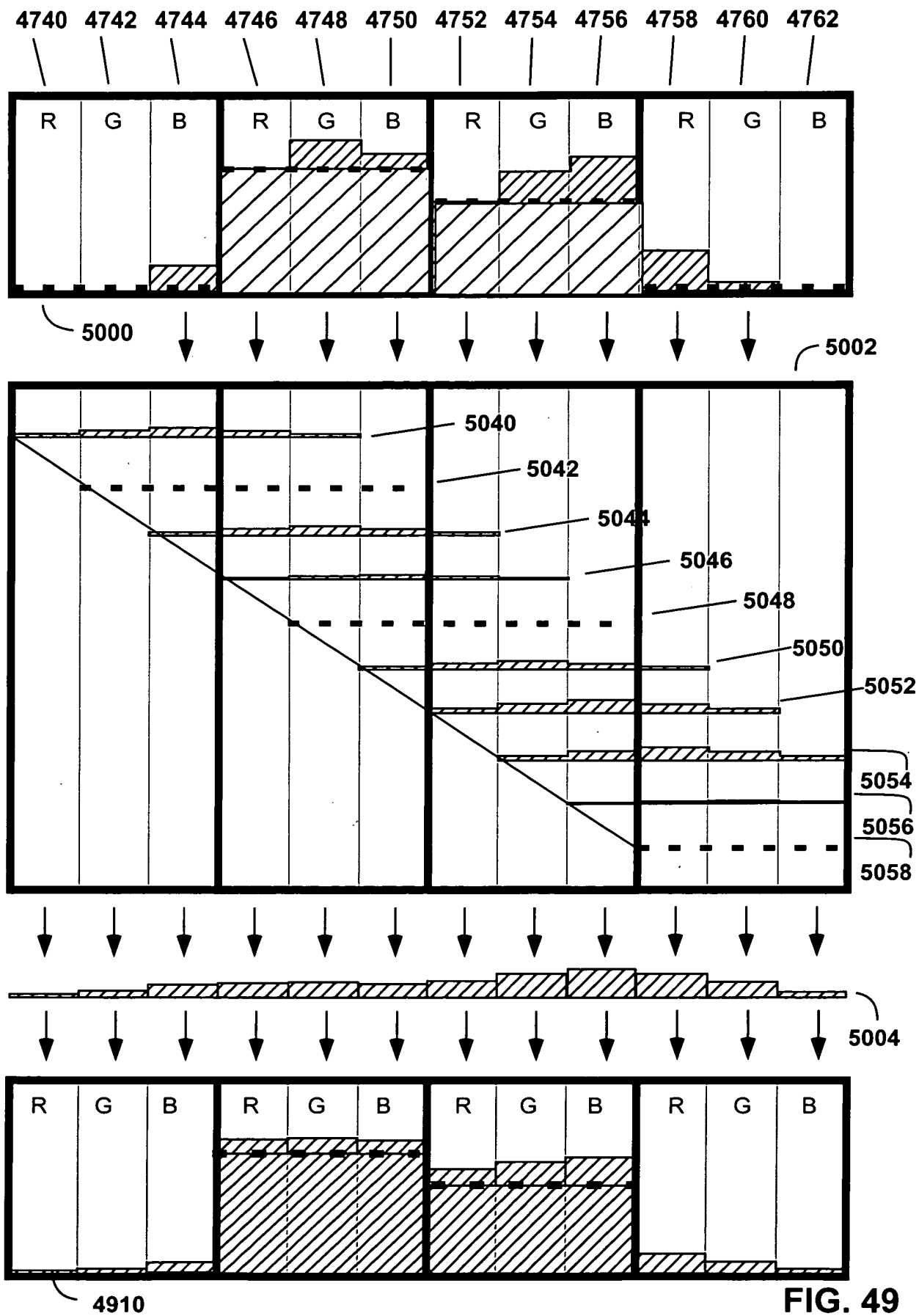
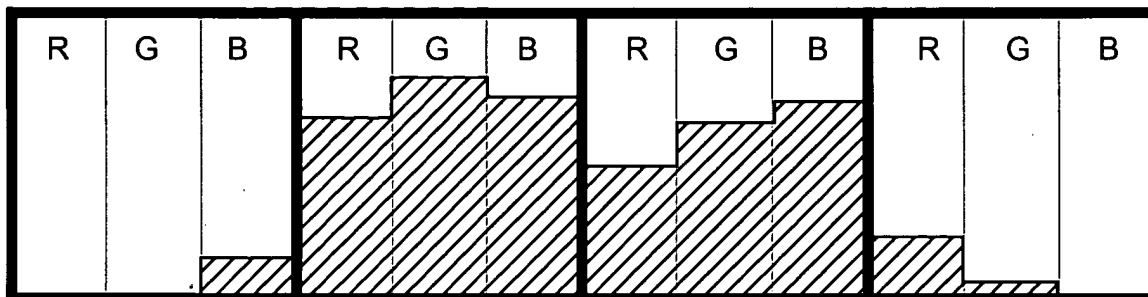
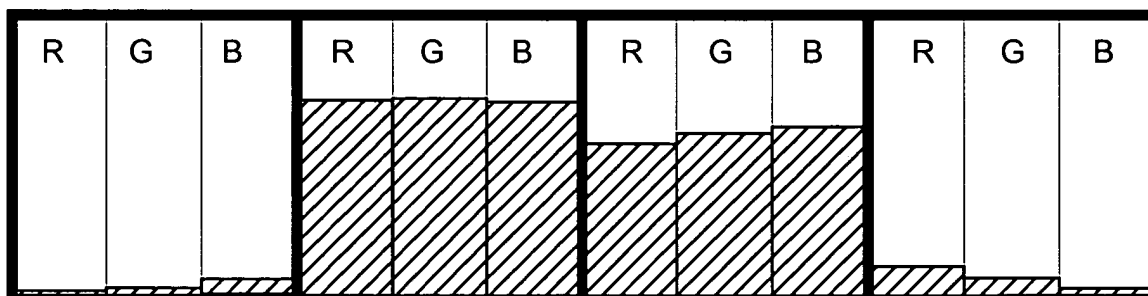


FIG. 48

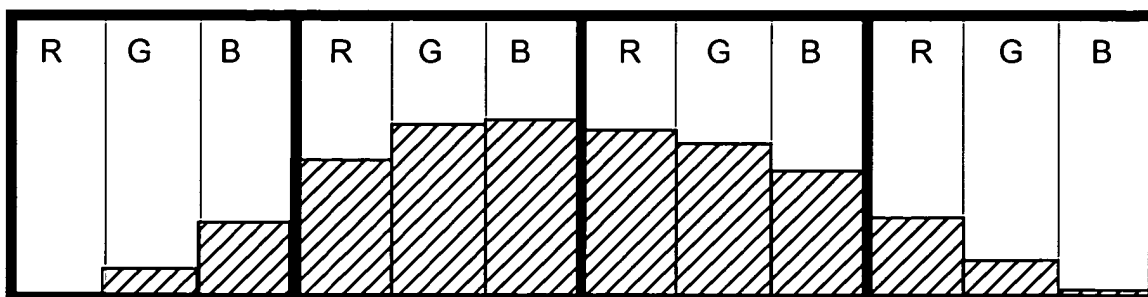




**FIG. 50**



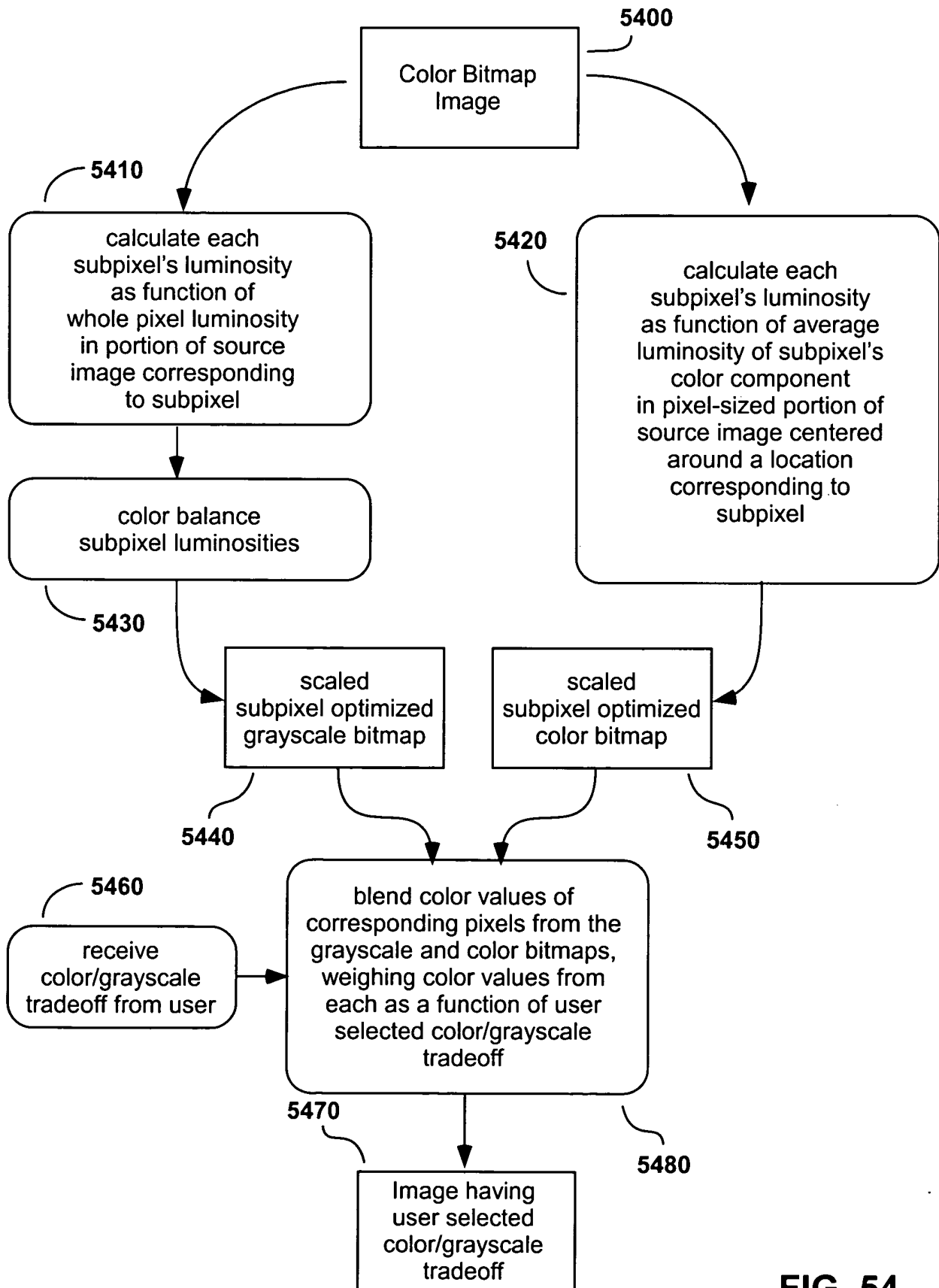
**FIG. 51**



**FIG. 52**

- subpixel optimization for bicolor bitmap ~5300
- for each pixel row in image~5301
  - for each pixel in row~5302
    - for each subpixel in the pixel~5304
      - determine which pixels of source image are in a window portion of source image corresponding to subpixel's area in scaled image~5306
      - for each source image pixel all or partially in the subpixel's source image window~5308
        - calculate the percent of the window's area covered by the pixel's area~5310
        - add to a luminosity/coverage value calculated for the subpixel the multiple of the percentage of the window area covered by the pixel times the pixel average foreground color intensity~5312
    - find the minimum subpixel luminosity/coverage value so calculated for the pixel~5314
    - for each subpixel in pixel~5316
      - set a luminosity/alpha value being calculated for the sub and the pixel to the pixel's minimum subpixel luminosity/coverage value~5318
      - distribute that portion of the subpixel's luminosity/coverage value that exceeds the pixel's minimum subpixel luminosity/coverage value to the luminosity/alpha values being calculated for the subpixel and adjacent subpixels in the pixel row using a color balance distribution filter~5320
  - For each pixel in row~5322
    - set the pixel's color value equal to a color having an compound RGB value with red, green, and blue component values equal to thlse luminosity/alpha values calculated for the pixel's red, green, and blue subpixels, respectively.~5324

**FIG. 53**



**FIG. 54**



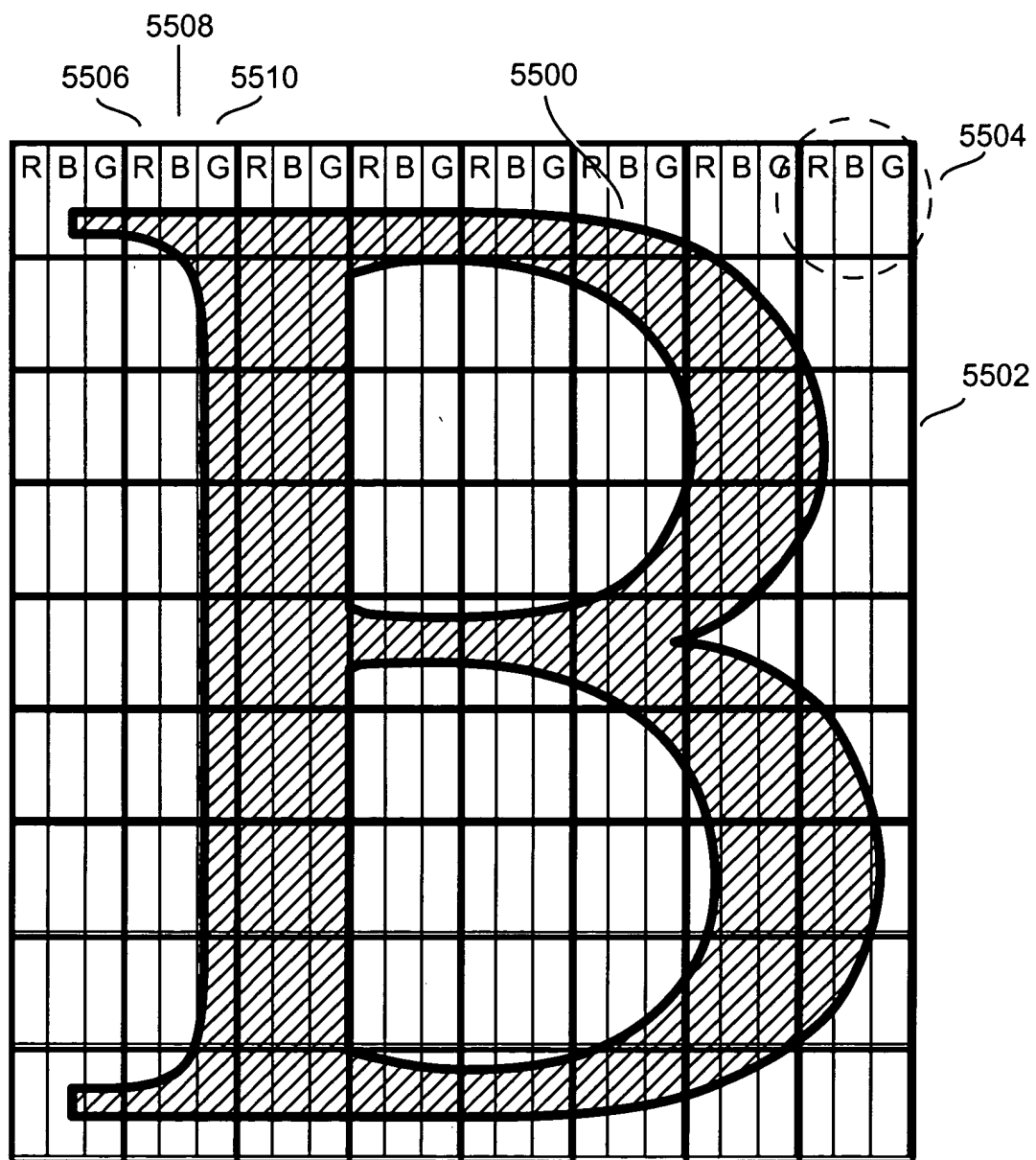


FIG. 55

## looksmart The quality web directory

The global leader in search infrastructure, LookSmart provides quality search solutions to leading portals, media companies and ISPs around the world.

For Businesses and Webmasters  
Submit your Web site to the LookSmart network and reach 83 percent of US Web users. Enable your site with Beseen.

For ISPs and Portals  
We provide search solutions for MSN,

► Search the Web

Search

► Explore by Topic [\[Submit a Site\]](#)

Entertainment  
Arts & Culture  
Celebrities Games  
Humor & Fun  
Movies Music  
Television

Shopping  
Auctions Automotive  
Buying Guides Cards  
& Gifts Classifieds  
Online Stores

People & Chat

Work & Money  
Business Companies  
Industries Jobs  
Personal Finance  
Professions Small Business

Computing  
Computer Science  
Multimedia  
Hardware Internet  
Networks & Communication  
Sales Software

Sports

**Get Listed Fast**  
Reach 83 percent of US Web users on MSN, Excite, AltaVista and other top search engines.

Express Submit  
Submit your site for review within two business days! guaranteed!

Subsite Listings  
Submit multiple links to deep content on your site.

Global Directories  
Australia  
Canada

FIG. 56

## looksmart The quality web directory

The global leader in search infrastructure, LookSmart provides quality search solutions to leading portals, media companies and ISPs around the world.

For Businesses and Webmasters  
Submit your Web site to the LookSmart network and reach 83 percent of US Web users. Enable your site with Beseen.

For ISPs and Portals  
We provide search solutions for MSN,

► Search the Web

Search

► Explore by Topic [\[Submit a Site\]](#)

Entertainment  
Arts & Culture  
Celebrities Games  
Humor & Fun  
Movies Music  
Television

Shopping  
Auctions Automotive  
Buying Guides Cards  
& Gifts Classifieds  
Online Stores

People & Chat

Work & Money  
Business Companies  
Industries Jobs  
Personal Finance  
Professions Small Business

Computing  
Computer Science  
Multimedia  
Hardware Internet  
Networks & Communication  
Sales Software

Sports

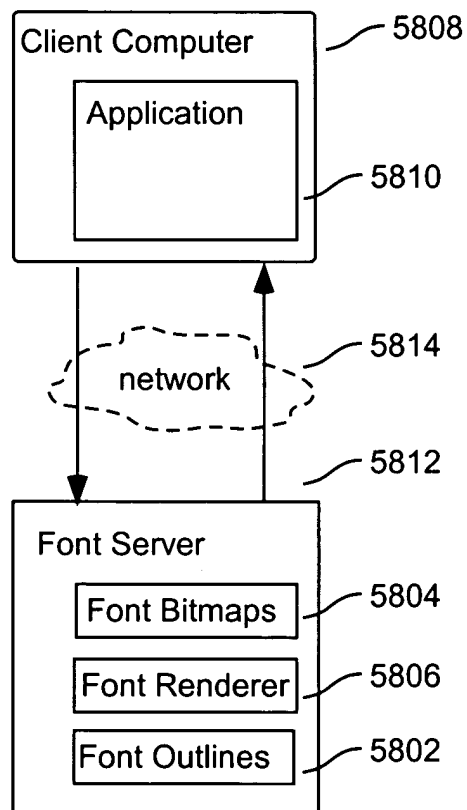
**Get Listed Fast**  
Reach 83 percent of US Web users on MSN, Excite, AltaVista and other top search engines.

Express Submit  
Submit your site for review within two business days! guaranteed!

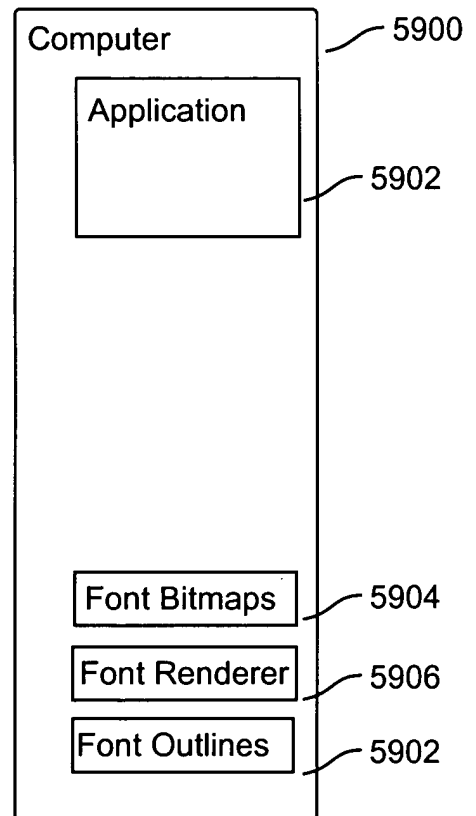
Subsite Listings  
Submit multiple links to deep content on your site.

Global Directories  
Australia  
Canada

FIG. 57



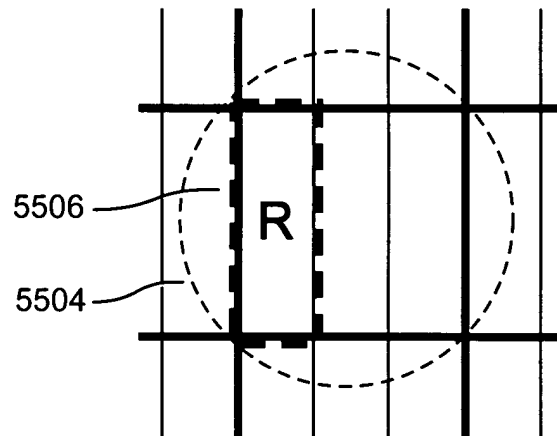
**FIG. 58**



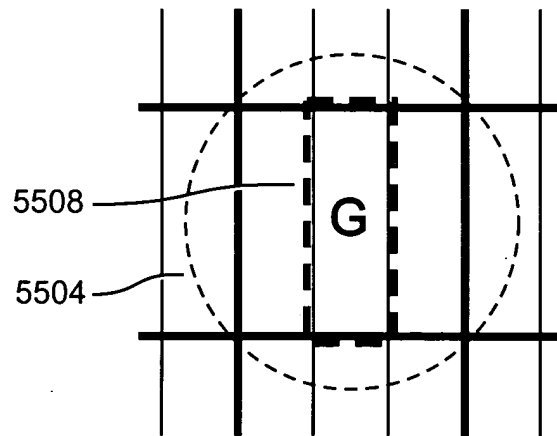
**FIG. 59**

- subpixel optimized font bitmaps with non-linear color balance~6000
  - for each pixel row~6002
    - for each subpixel in a row~6004
      - determine a coverage value representing the percent the subpixel which is covered by the font shape~6006
    - for each pixel in row~6008
      - determine the minimum coverage value calculated for each of its three subpixels ~6010
      - add minimum to temporary alpha value being calculated for each subpixel of the pixel~6012
      - for each of pixel's subpixels~6014
        - determine excess of subpixel's coverage value over the pixel's minimum~6016
        - distribute this excess value into sub-pixel alpha values based on color balance distribution filter appropriate for sub-pixel's color~6018
  - for each pixel in row~6020
    - use the three color alpha value defined for each pixel by the three alpha values calculated for its subpixels in a look-up table to map that value into one of relatively small palette of colors~6022

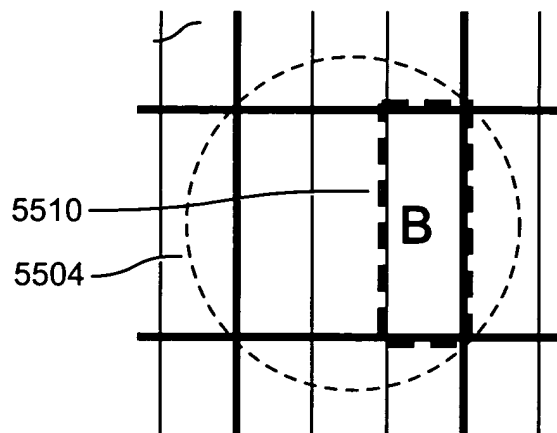
## **FIG. 60**



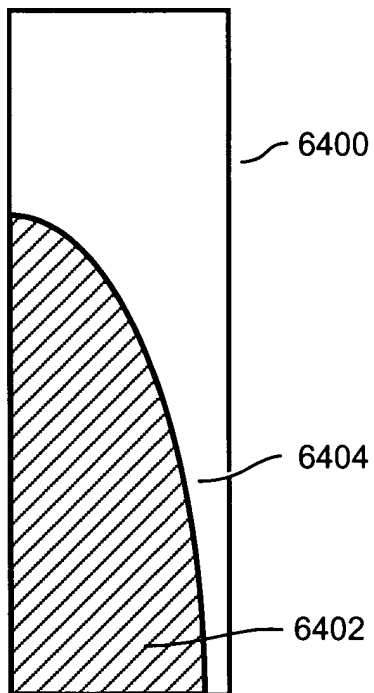
**FIG. 61**



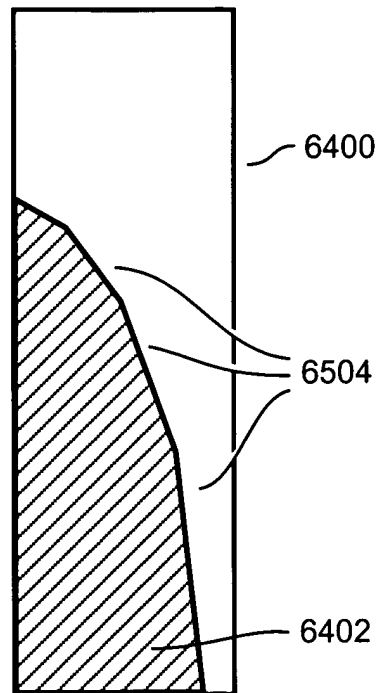
**FIG. 62**



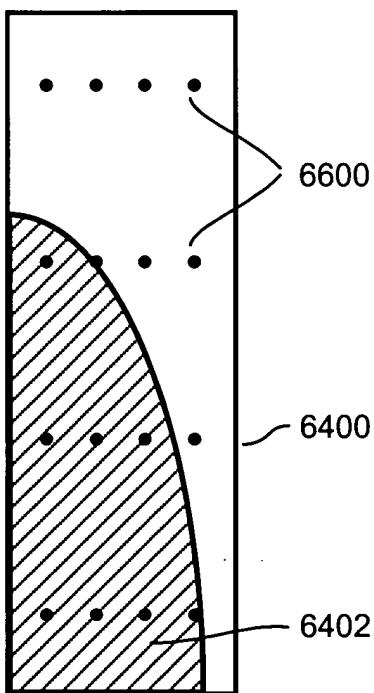
**FIG. 63**



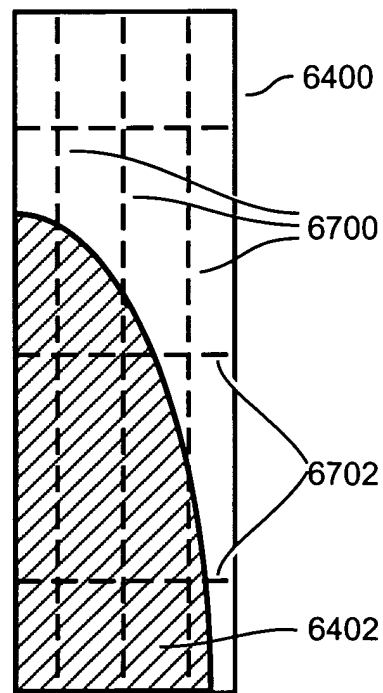
**PRIOR ART  
FIG. 64**



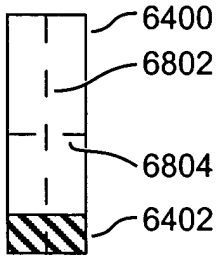
**PRIOR ART  
FIG. 65**



**PRIOR ART  
FIG. 66**



**PRIOR ART  
FIG. 67**



**FIG. 68**



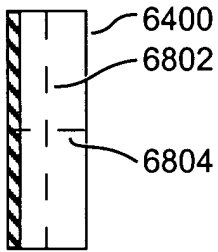
**FIG. 69**



**FIG. 70**



**FIG. 71**



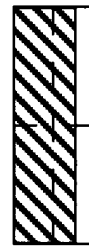
**FIG. 72**



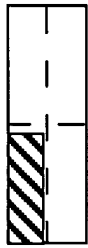
**FIG. 73**



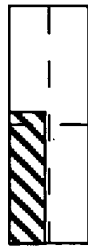
**FIG. 74**



**FIG. 75**



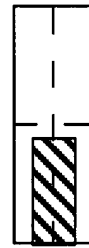
**FIG. 76**



**FIG. 77**



**FIG. 78**



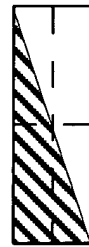
**FIG. 79**



**FIG. 80**



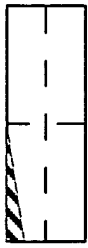
**FIG. 81**



**FIG. 82**



**FIG. 83**



**FIG. 84**



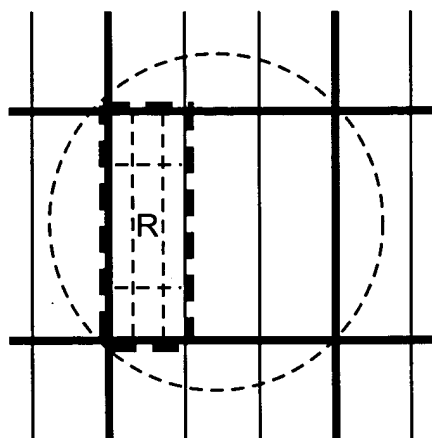
**FIG. 85**



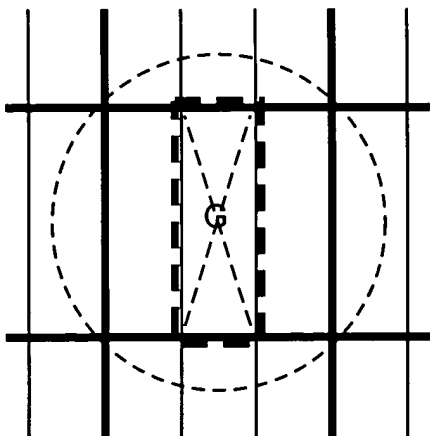
**FIG. 86**



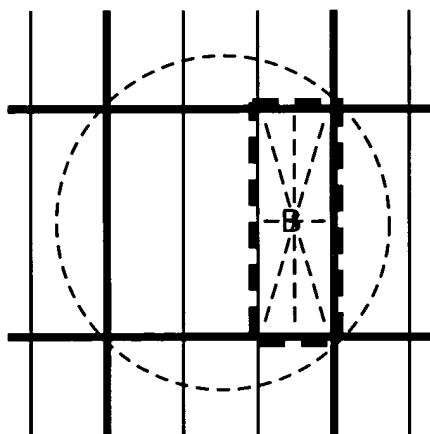
**FIG. 87**



**FIG. 88**

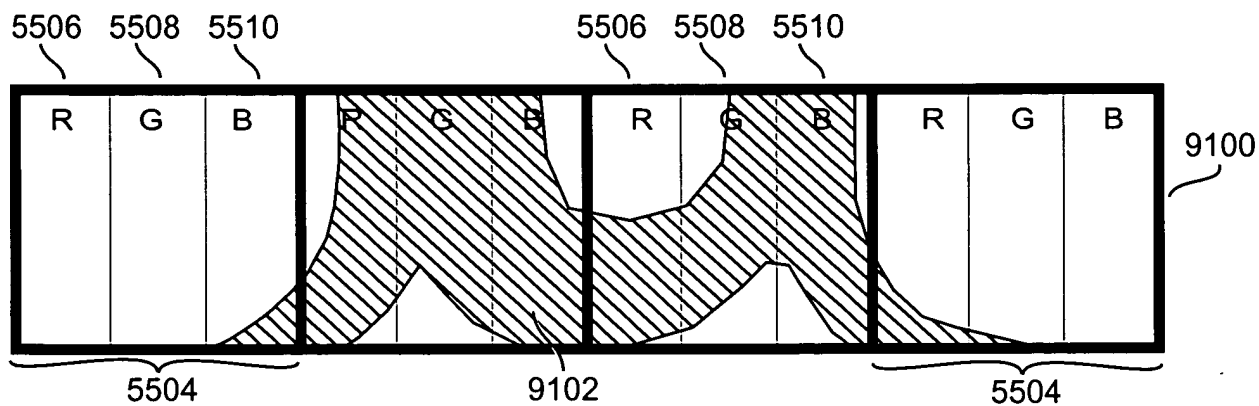


**FIG. 89**

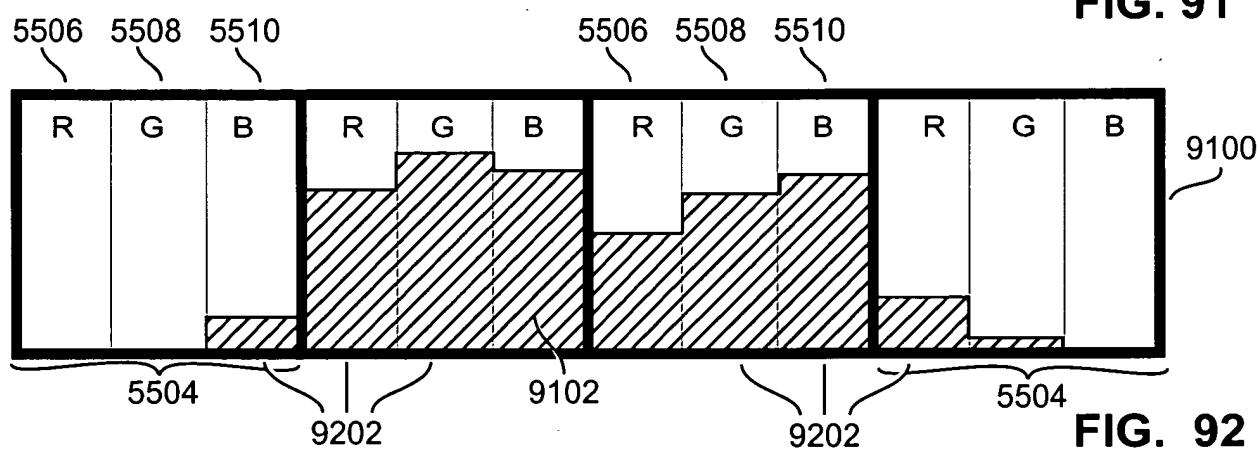


**FIG. 90**

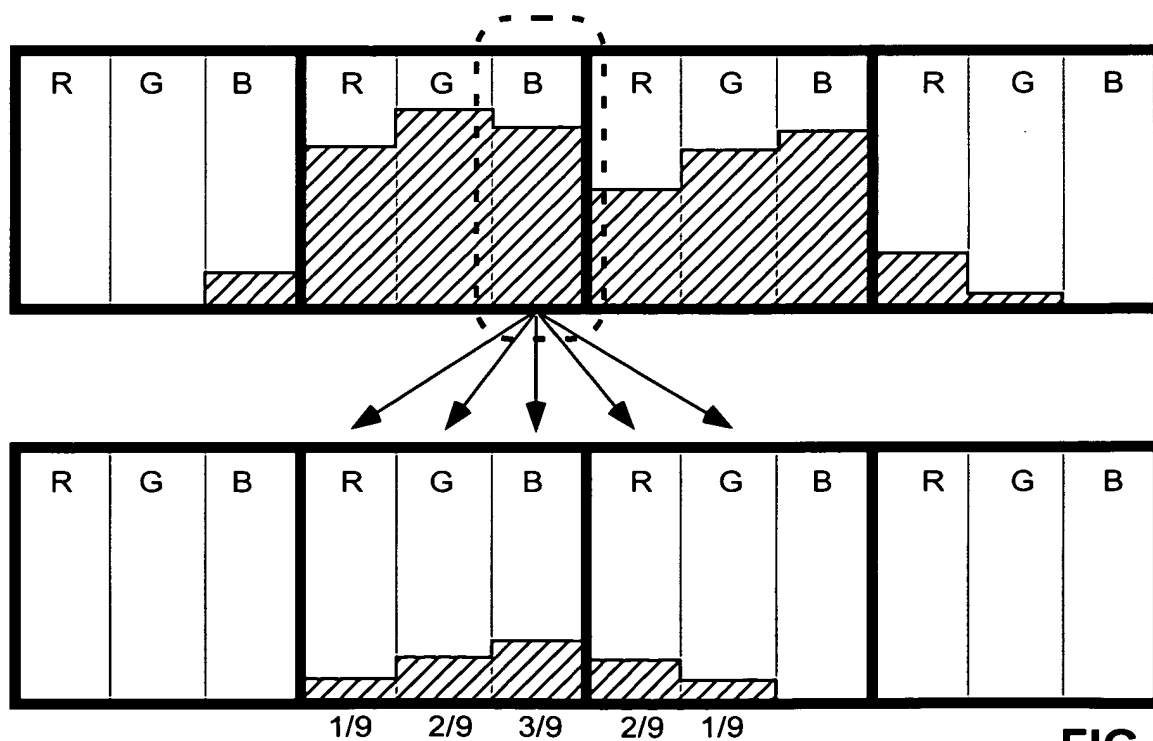




**FIG. 91**



**FIG. 92**



**FIG. 93**

-Center-Weighted Color Balance Filter~9400

-For Coverage Value 0	0,0,0,0,0
-For Coverage Value 31	0,1,1,1,0
-For Coverage Value 62	0,2,2,2,0
-For Coverage Value 93	1,2,3,2,1
-For Coverage Value 126	1,3,4,3,1

**FIG. 94**

-Asymetric Color Balance Filter~9500

-For Coverage Value 0	0,0,0,0,0
-For Coverage Value 31	1,1,1,0,0
-For Coverage Value 62	2,2,2,0,0
-For Coverage Value 93	3,3,3,0,0
-For Coverage Value 126	3,3,4,1,1

**FIG. 95**

- Creation of input-color-to-output-color look-up table~9600
  - run the characters of multiple fonts through non-linear algorithm for deriving subpixel optimized font bitmaps, while keeping a histogram of the number of times each of one of the 2196 possible three color alpha values is calculated for a pixel~9602
  - create a 122 color output palette by~9604
    - selecting the thirteen grayscale colors possible for whole pixel alpha values in which each subpixel can have one of thirteen levels including, black and white~9606
    - selecting the 109 other most frequently occurring colors in the histogram~9608
  - for each of the 2196 possible calculated alpha values~9610
    - if that input color exactly matches one of the palette's colors~9612
      - associate the input color with that palette, or output, color~9614
    - if not~9616
      - for each of 122 output colors~9618
        - if (ri-ro) and (gi-go) are of same sign and if  $|ro-go| < |ri-gi|+x$ ~9620
          - calculate the distance from the input color to the output color~9622
          - if that distance is the closest distance so far to the input color~9624
            - save it as closestAllowedPaletteColor~9626
    - associate the input color with the closestAllowedPaletteColor~9628

**FIG. 96**

- Displaying text with font bitmaps having subpixel alpha value components ~9700
  - for each string to be displayed~9702
    - sample points in rectangle of bitmap in which it is to be drawn to determine its background color~9704
    - for each of the 122 whole pixel alpha values used to represent font bitmaps~9706
      - for each of the three subpixel component colors~9708
        - calculate color luminosity for the subpixel component color as the current subpixel's alpha value in the current whole pixel alpha value ( the current supixel's alpha) times luminosity of the current subpixel color in the foreground color with which the fonts are to be drawn plus (1 minus the current subpixel's alpha) times the luminosity of the current subpixel's color in the background color~9710
      - map the current whole pixel alpha value to the whole pixel color comprised of the three subpixel luminosities calculated for the whole pixel alpha value~9712
  - for each character of string to be displayed~9714
    - access its associated font bitmap~9716
    - for each of pixel of the font bitmap~9718
      - find the color value which has been mapped to the pixel's corresponding whole pixel alpha value in the font bitmap~9720
      - set the corresponding pixel in the subpixel addressable display to that whole pixel color value~9722

**FIG. 97**

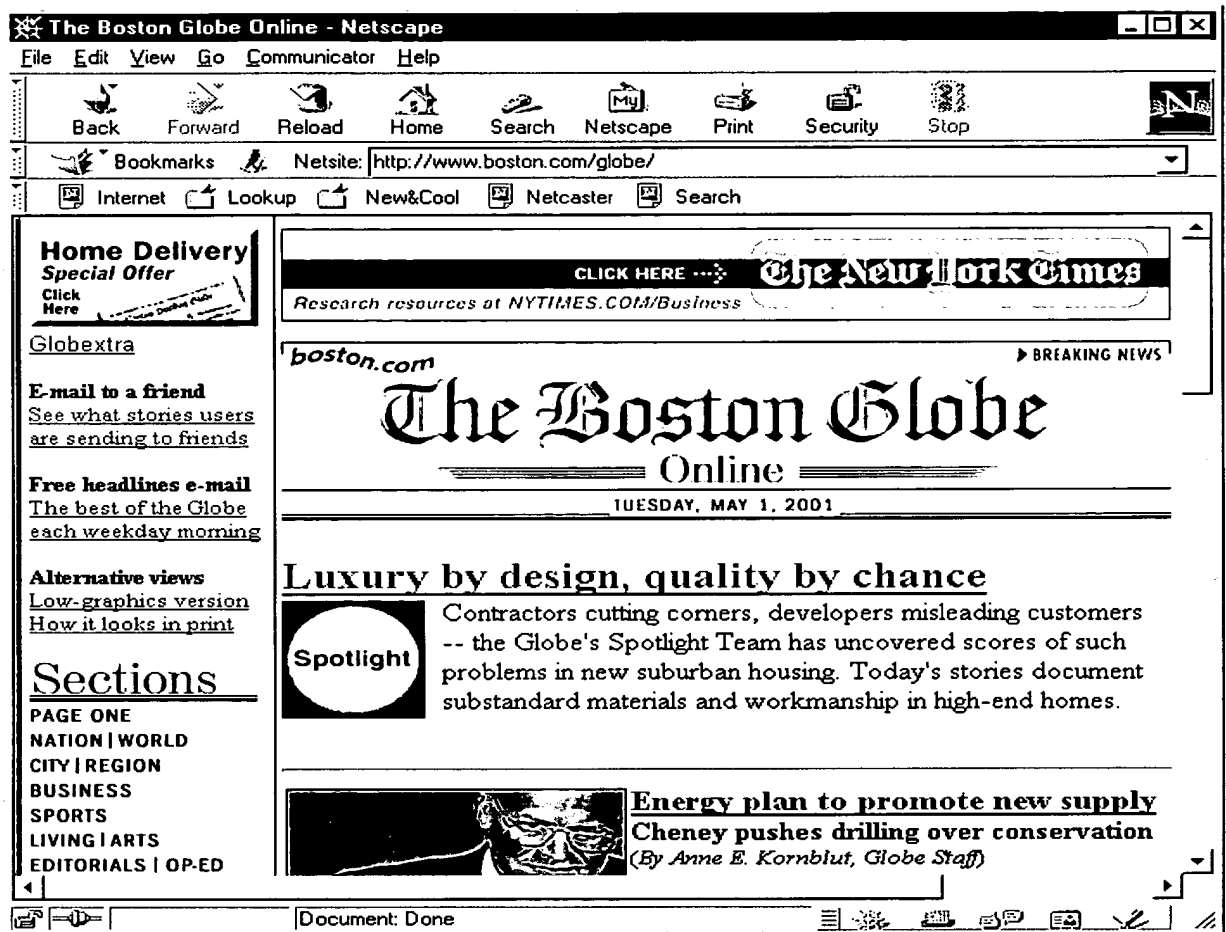


FIG. 98

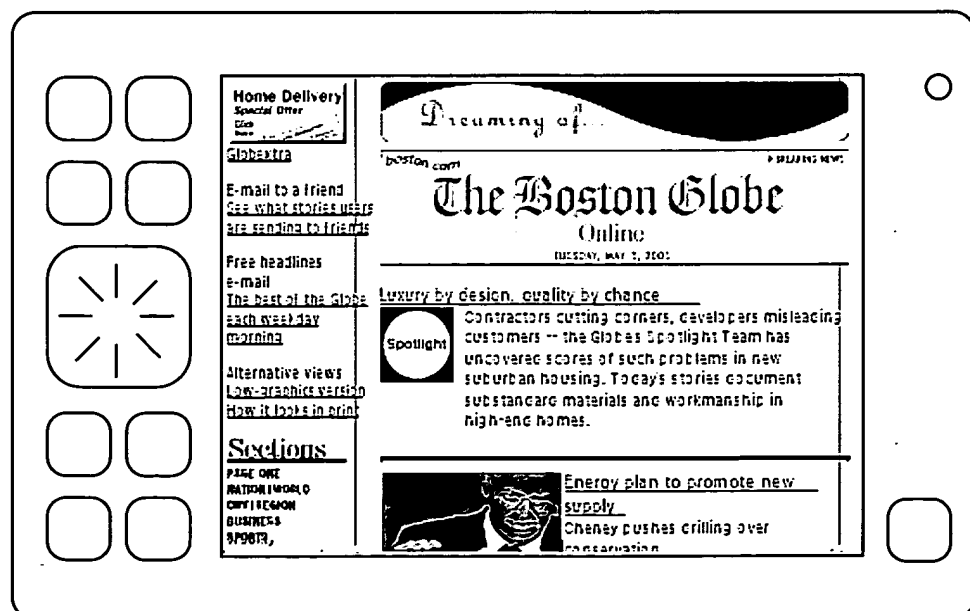


FIG. 99

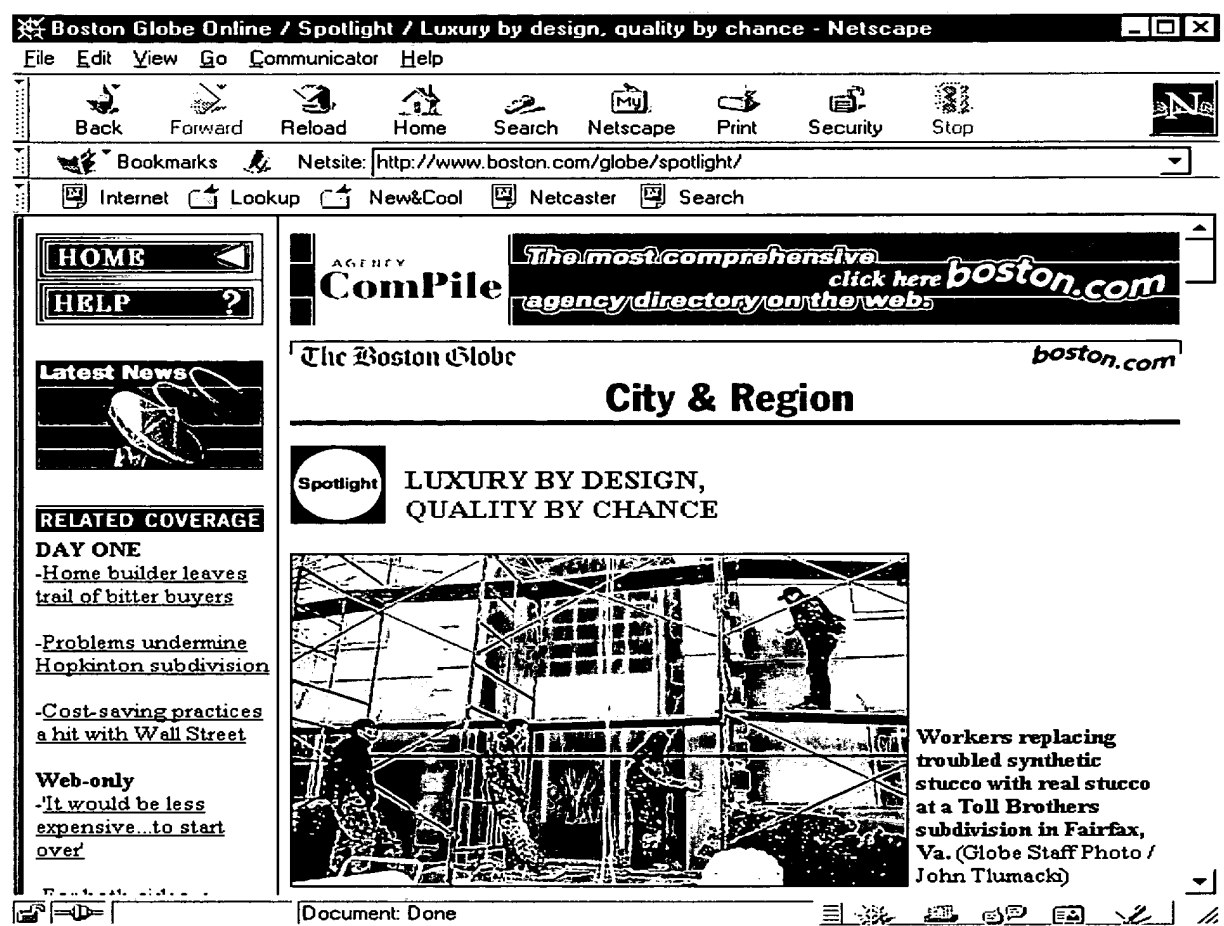


FIG. 100

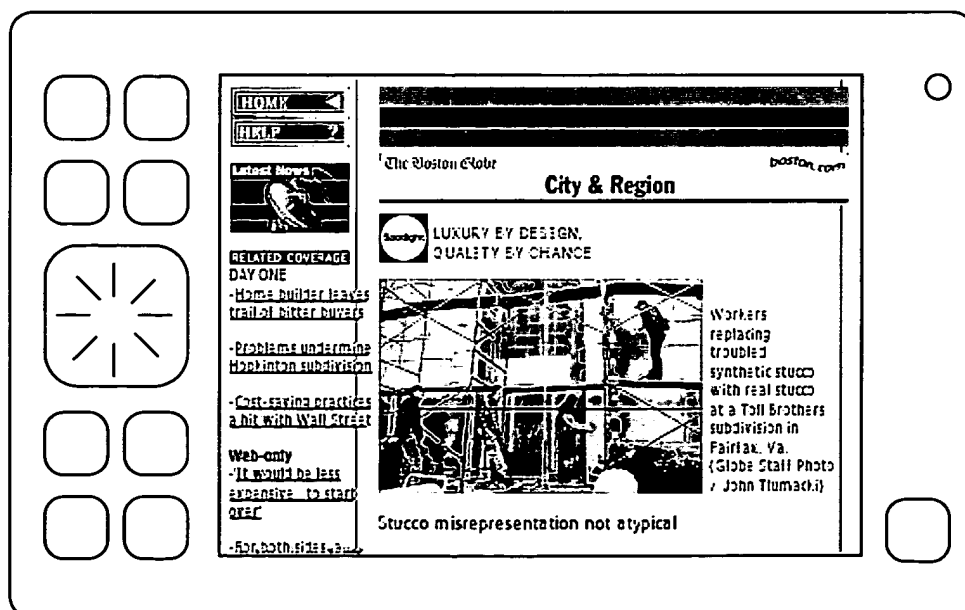
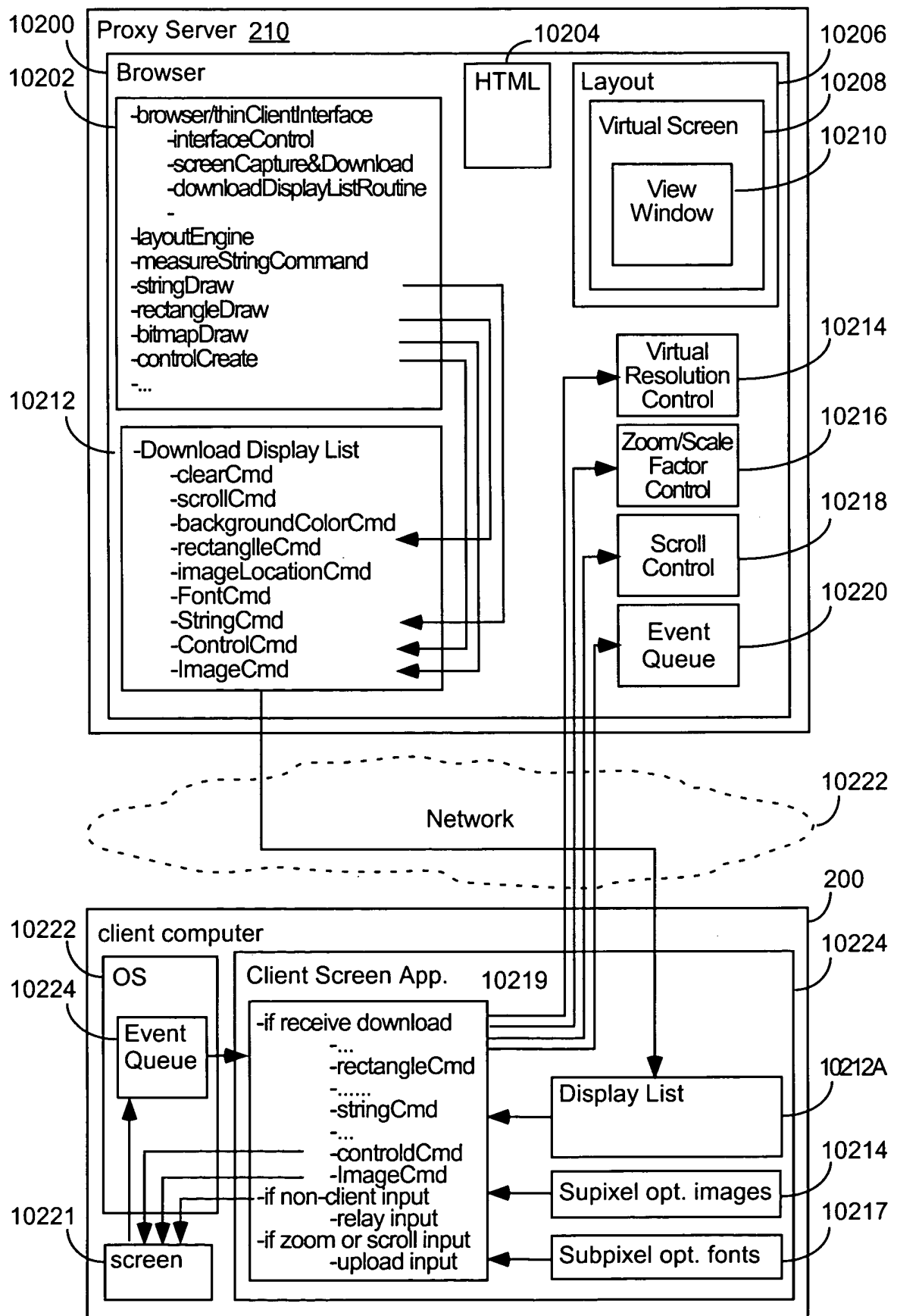


FIG. 101



**FIG. 102**

-Sample html~10300

```
-</head>

-<body
background="http://a1636.g.akamai.net/7/1636/797/e5e77dd148cc98/graphics.
boston.com/globe/images/tiles/tile.gif" link="5C3317" vlink="5C4033"
onLoad="FrameThis()">
-...
-...
-<p>
    -<B>E-mail to a friend</b><BR>
    -<a href="http://tools.boston.com/pass-it-on/popular/">See what stories users
    are sending to friends</a><p>
    -<B>Free headlines e-mail</B><br>
    -<a href="/newsletters/headlines/">The best of the Globe each weekday
    morning</a><p>
    -<B>Alternative views</b><BR>
    -<a href="/globe/lowgraphics">Low-graphics version</a><br>
    -<a href="/globe/acrobat/today.pdf">How it looks in print</a>
    -<p>
    -<!--SECTIONS-->
    -<IMG WIDTH="120" HEIGHT="27"
    SRC="http://a1636.g.akamai.net/7/1636/797/0df5e88d0bb528/graphics.boston.
    com/globe/images/navs/nsections.gif" border="0" alt="Sections" VSPACE="1"
    WIDTH="120" HEIGHT="27"><BR>
    -<a href="/globe" onMouseOver="loadimage('pageone',
    'http://graphics.boston.com/globe/images/navs/rpageone.gif'); status='Boston
    Globe Online: Page One'; return true;" onMouseOut="loadimage('pageone',
    'http://graphics.boston.com/globe/images/navs/npageone.gif'); status=''; return
    true;"><IMG WIDTH="119" HEIGHT="14"
    SRC="http://a1636.g.akamai.net/7/1636/797/f0a63d528dc7e3/graphics.boston.
    com/globe/images/navs/npageone.gif" border="0" alt="Boston Globe Online:
    Page One" name="pageone" VSPACE="1"></a><br>
    -...
    -...
```

FIG. 103



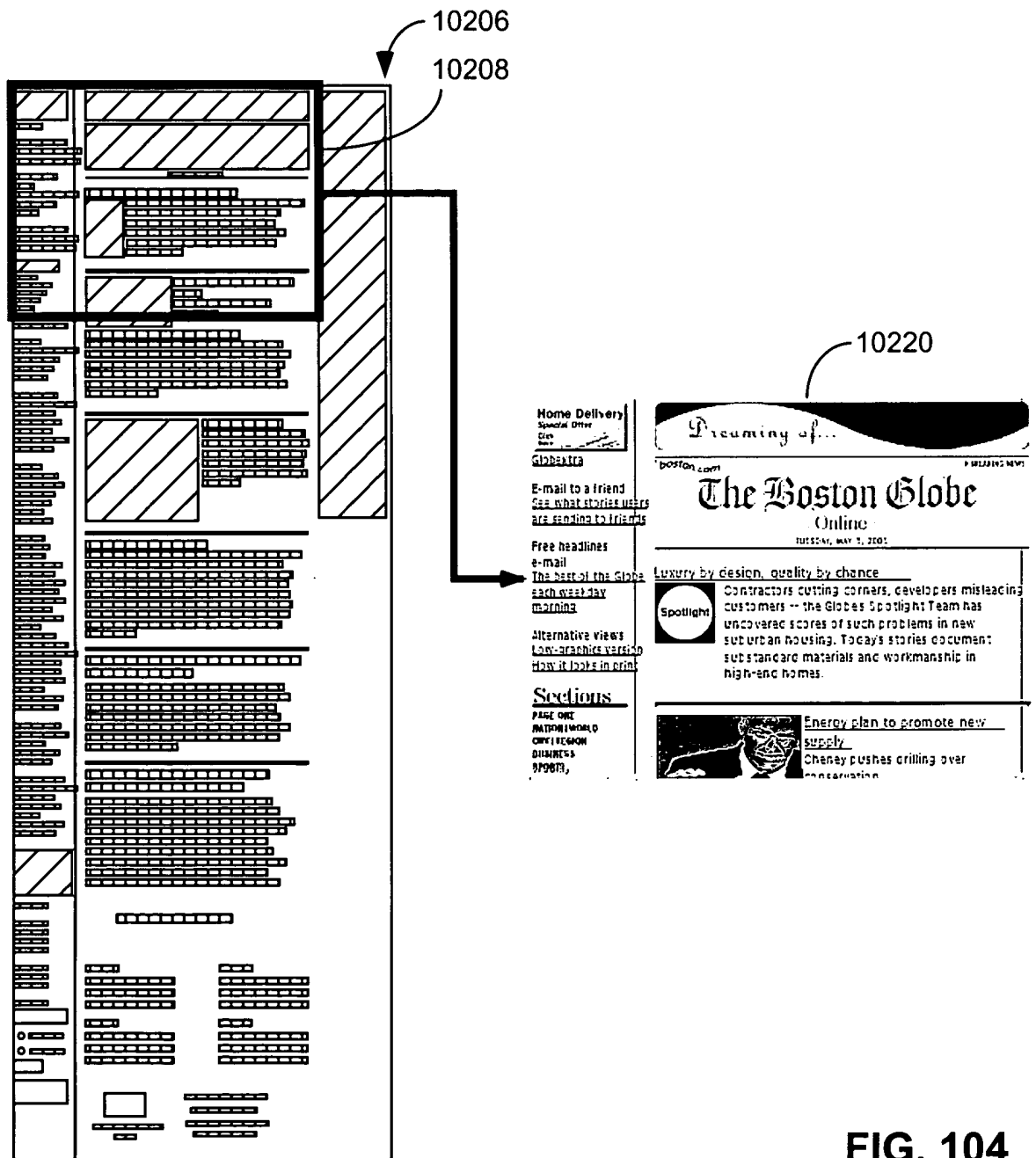


FIG. 104

-Browser's Proxy Code~10500

-...

- if receive request from thin client for a web page~10502
  - relay request to server indicated in URL of request~10504
- if receive indication that browser has completed a screen draw or redraw~10506
  - call the screen capture and download routine when screen redraw is complete ~10510
- if receive control object state query from browser~10514
  - query thin client for state of indicated one or more controls~10516
  - send those control states to browser~10518
- if receive scroll/move command from thin client~10520
  - move view window accordingly relative to browser's virtual screen~10522
  - if portion of view window which was in view window before move is still in view window~10526
    - place appropriate scroll command at start of download display list~10528
  - if moved view window includes portion of web page not currently in virtual screen~10530
    - scroll browser's virtual screen accordingly~10532
    - request redraw for the newly scrolled virtual screen from browser~10534
- if receive zoom command from thin client~10536
  - change view scale factor accordingly~10538
  - scale view window accordingly relative to browser's virtual screen~10540
  - if scaled view window includes portion of web page not currently in virtual screen~10542
    - scroll virtual screen change its resolution to cause scaled view window to fit in virtual screen~10544
  - call for screen redraw~10552

**FIG. 105A**

- if receive virtual resolution command from thin client~10554
  - change browser's virtual screen resolution to requested virtual resolution~10556
  - call for screen redraw~10560
- if receive other user input event from thin client~10562
  - transform its screenXY on client screen to corresponding location in browser screen using location of view window and display scaling factor~10564
  - relay event to browser's event queue~10566
- ...

**FIG. 105B**

- Screen Capture And Download Routine~10600
  - ask for browser for screen redraw~10602
  - if browser calls~10604
    - measureString~10606
      - map requested font family and font size into substitute font family and size, including:~10608
        - select size for substitute fonts as a function of the requested font size and downscaling by the display scaling factor~10610
        - replace smaller size fonts with narrower and taller size to take advantage of the higher ratio of horizontal resolution of subpixels~10612
        - if limitMinimumFontSize is on~10614
          - prevent substitute font size from being below a minimum pixel size~10616
      - return string measurement for substituted font and font size, scaled up measurements by display scale factor~10618
  - stringDraw~10620
    - transform string's virtual screen screenXY to thin client screenXY by scaling and/or translating as a function of the display scale factor and the view window's position relative to virtual screen~10621
    - if substituted font family and size associated with string in prior measure string call and any other font attributes for string is different than current font attribute for end of display list~10622
      - store a font command at end of display list changing current font attributes to attributes corresponding to current string~10623
    - store string, its thin-client screenXY, and its substituted font, including size and color, at end of download display list~10624
  - rectangleDraw~10626
    - transform rectangle's virtual screen screenXY, width, and height to thin client screenXY, width and height by scaling and/or translating as a function of the display scale factor and the view window's position relative to virtual screen~10628
    - if rectangle's color is different than background color for current end of download display list~10630
      - add a background color command changing to new background color at end of download display list~10632
    - store rectangle, transformed screenXY, width, and height at end of download display list~10634

**FIG. 106A**

- bitmapDraw~10636
  - if images URL is not already in a download image list~10638
    - if bitmap is color bitmap~10642
      - scan image for one or more portions of sufficient size which have only colors from a given bicolor spectrum~10644
      - for each bicolor portion of image found~10646
        - perform bicolor subpixel optimization, scaled down by display scale factor, on portion using opposite ends of its bicolor spectrum as foreground and background color at display scale factor~10648
        - if foreground color is too chromatically unbalanced, render image with a substituted more-balanced foreground color~10650
      - for each non-bicolor portion of image found~10652
        - perform multicolor subpixel optimization, scaled down by display scale factor, on bitmap using color image subpixel algorithm at display scale factor~10654
    - else if bitmap is grayscale bitmap~10656
      - perform bicolor subpixel optimization, scaled down by display scale factor, on bitmap using black and white as foreground and background colors at display scale factor~10658
    - store scaled-down, subpixel-optimized bitmap, with a unique imageID, transformed width and height, and its URL at end of image list~10662
  - transform image's screenXY for download and store an image location command having the imageID and the transformed screenXY, width, and height stored for the image in the image list at end of download display list~10664

**FIG. 106B**

- controlCreate~10666
  - transform control's screenXY by scaling and/or translating as a function of the display scale factor and the view window's position relative to virtual screen~10667
  - place corresponding control command, its transformed screenXY, and corresponding text in download list~10668
  - create corresponding browser-side portion of distributed control~10670
- ...
- when screen redraw is complete~10672
  - call download display list routine~10674
  - clear display list~10676

**FIG. 106C**

- download display list routine~10700
  - select all elements in display list which will be all or partially be included in the new image which is to be created on the thin client's display screen in a download stream~10702
  - place all bitmaps in image list corresponding to one or more image locations commands in download stream at end of download stream, performing lossy compression on them first~10704
  - user a lossless compression algorithm to compress download stream~10705
  - open a the socket connection between browser's computer and thin client~10706
  - send display list to thin client over socket connection~10708

**FIG. 107**

-Download Stream~10800

-clearCmd~10802

-scrollCmd + XYShift~10804

-...

-backgroundColorCmd + color~10806

-rectangleCmd +ScreenXY + width + Height~10808

-rectangleCmd +ScreenXY + width + Height~10808

-...

-backgroundColorCmd + color~10806

-imageLocationCmd + ImageID + ScreenXY+ width + height~10810

-FontCmd + FontAttribute1 + NewValue1 + FontAttribute2 +  
NewValue2~10812...

-StringCmd + ScreenXY + String~10814

-StringCmd + ScreenXY + String~10814

-...

-imageLocationCmd + ImageID + ScreenXY+ width + height~10810

-FontCmd + FontAttribute3+ NewValue3...~10812

-StringCmd + ScreenXY + String~10814

-...

-StringCmd + ScreenXY + String~10814

-StringCmd + ScreenXY + String~10814

-rectangleCmd +ScreenXY + width + Height~10808

-ControlCmd + ControlID + ControlType + ScreenXY + ControlTextList~10816

-StringCmd + ScreenXY + String~10814

-ControlCmd + ControlID + ScreenXY + Control Label~10816

-ImageCmd + ImageID + width + height + Bitmap~10818

-ImageCmd + ImageID + width + height + Bitmap~10818

-ImageCmd + ImageID + width + height + Bitmap~10818

**FIG. 108**

-thin client code~10900

-...

-if receive a download stream, start responding to individual commands in stream in the order in which they are received as soon as one or more are received, including responding to each of the following commands as follows~10902

-clearCmd ~10904

-clear thin client screen~10906

-scrollCmd~10908

-bitblit portion of screen which remains on screen after XYShift to appropriate position after that shift~10910

-clear rest of screen~10912

-backgroundColorCmd~10914

-set current rectangle background color to the color specified in command~10916

-rectangleCmd~10918

-draw rectangle with upper left hand corner at ScreenXY, having the width and height specified in command, using current background color~10920

-imageLocationCmd~10922

-do nothing~10923

-FontCmd + FontAttributer1 + NewValue1 + FontAttribute2 + NewValue2...~10924

-set current value of all font attributes listed in command to the corresponding values listed in the command~10926

-StringCmd + ScreenXY + String~10928

-if thin client does not have font bitmap for each character of the specified string at the current font size and font family~10930

-send separate HTTP request for font of each such bitmaps from font server, specifying size, character, font, and that is to be subpixel optimized, and subpixel array type~10932

-when receive each requested font~10934

-place it in font bitmap cache~10936

-if have all characters of string specified in the command~10938

-draw string, using current font attributes values, including foreground color, and using color from portion of screen on which it is being written as the background color~10940

**FIG. 109A**



- ControlCmd + ControlID + ControlType + ScreenXY + ControlTextList~10942
  - if no control has been created having controlID specified in command~10944
    - create thin client side of distributed control associated with that controlID~10946
  - draw specified control type on screen using subpixel optimized bitmaps for control image, at specified screenXY, using drawing one or more text items in controlTextList using subpixel optimized text, and set control's associated screen hotzone, if any to the appropriate portion of screen~10948
- ImageCmd~10950
  - for each imageLocationCmds in display list having same ImageID~10952
    - draw bitmap at that location~10954
  - redraw all other items in display list which occur at the same location as any of these drawn bitmaps~10956
- ~...
- if user clicks hotzone associated with text entry field~10958
  - execute keyboard mode routine~10960
    - display pop up user keyboard and text edit line, saving the bitmap which was in its location~10962
    - until user presses enter~10964
      - if user user types text character~10966
        - place corresponding subpixel-optimized text shape on text edit line~10968
        - add character to position corresponding to cursor in a temporary text-edit string variable~10970
  - ~...
  - when user press enter, ~10972
    - store text-edit string state in corresponding text field control~10974
    - draw text-edit string in bitmap of text entry field using subpixel optimized fonts~10976
    - remove popup keyboard, replacing the bitmap with was in its place before it was displayed~10978
- else if user clicks on hot zone of a button or menu item control~10980
  - change appearance of button or menu item appropriately~10981
  - send event along with button or menu item ID up to proxy~10982
- else if user clicks on hot zone of another type of thin client control~10983
  - change appearance of control appropriately~10984
  - store user selected state change~10985
- else if user clicks on other portion of screen not associated with the thin client program's or its computer's control interface~10986
  - send even with screen location up to proxy~10987

**FIG. 109B**

- if receive query from proxy process re state of control~10988
  - query state of corresponding control on thin client~10989
  - transmit state to browser~10990
- if user enters a command to scroll screen~10991
  - upload command to proxy~10992
- if user enters a command to change zoom,~10993
  - upload command to proxy~10994
- if user enters a command to change virtual resolution,~10995
  - upload command to proxy~10996
- if user enters another command associated with thin client's control GUI~10997
  - ...~10998

**FIG. 109C**

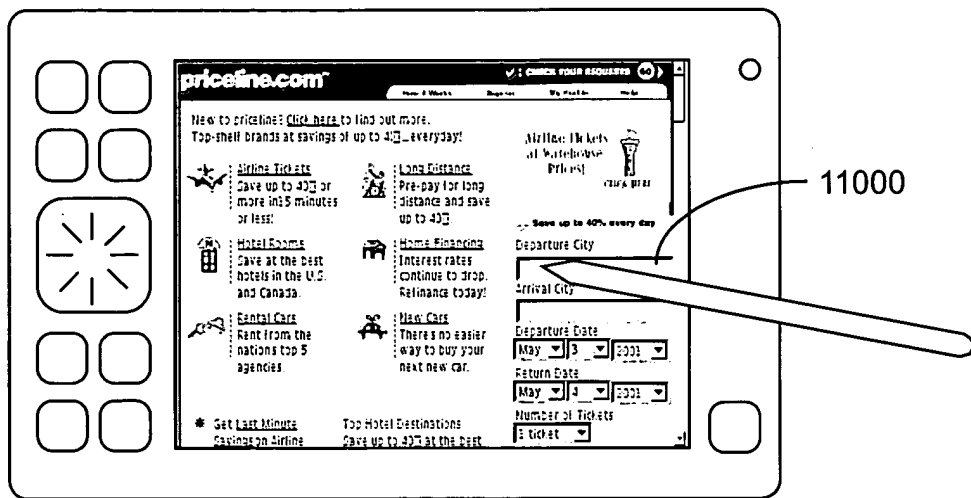


FIG. 110

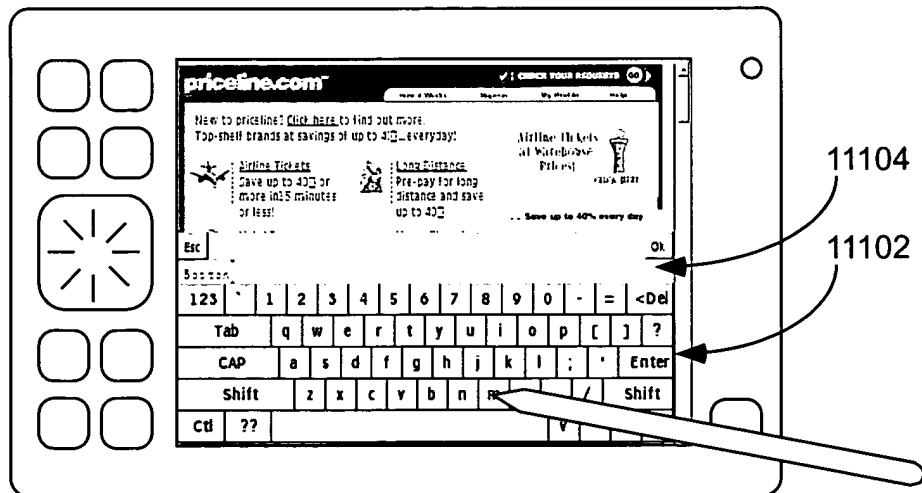


FIG. 111

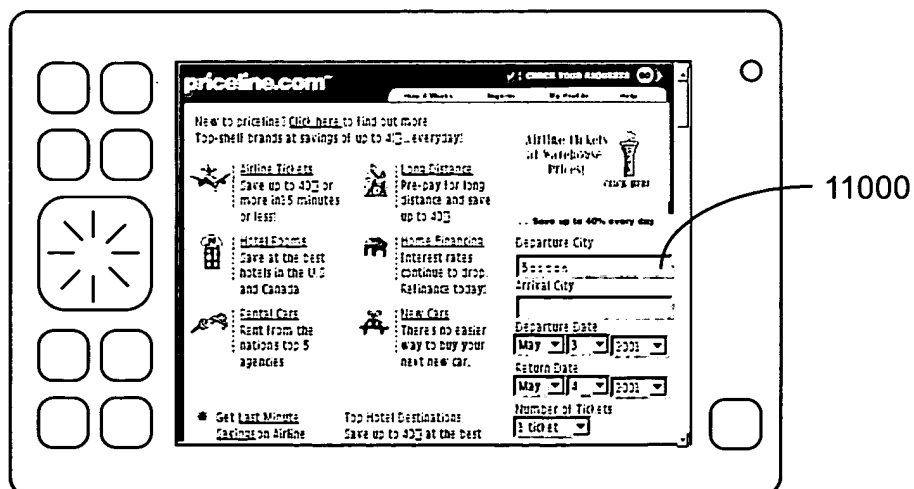


FIG. 112

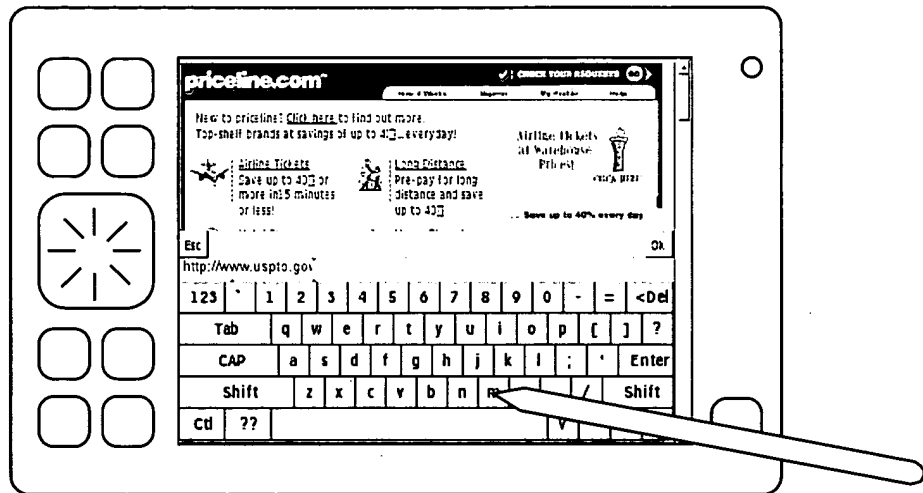


FIG. 113

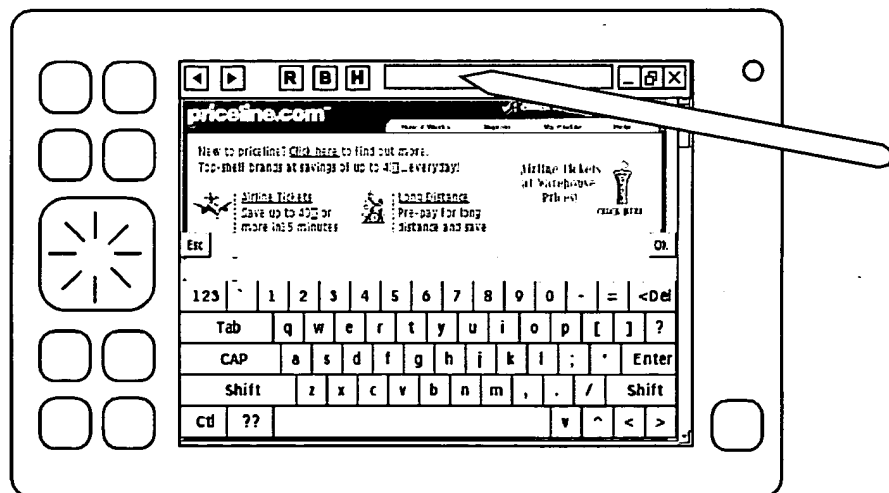


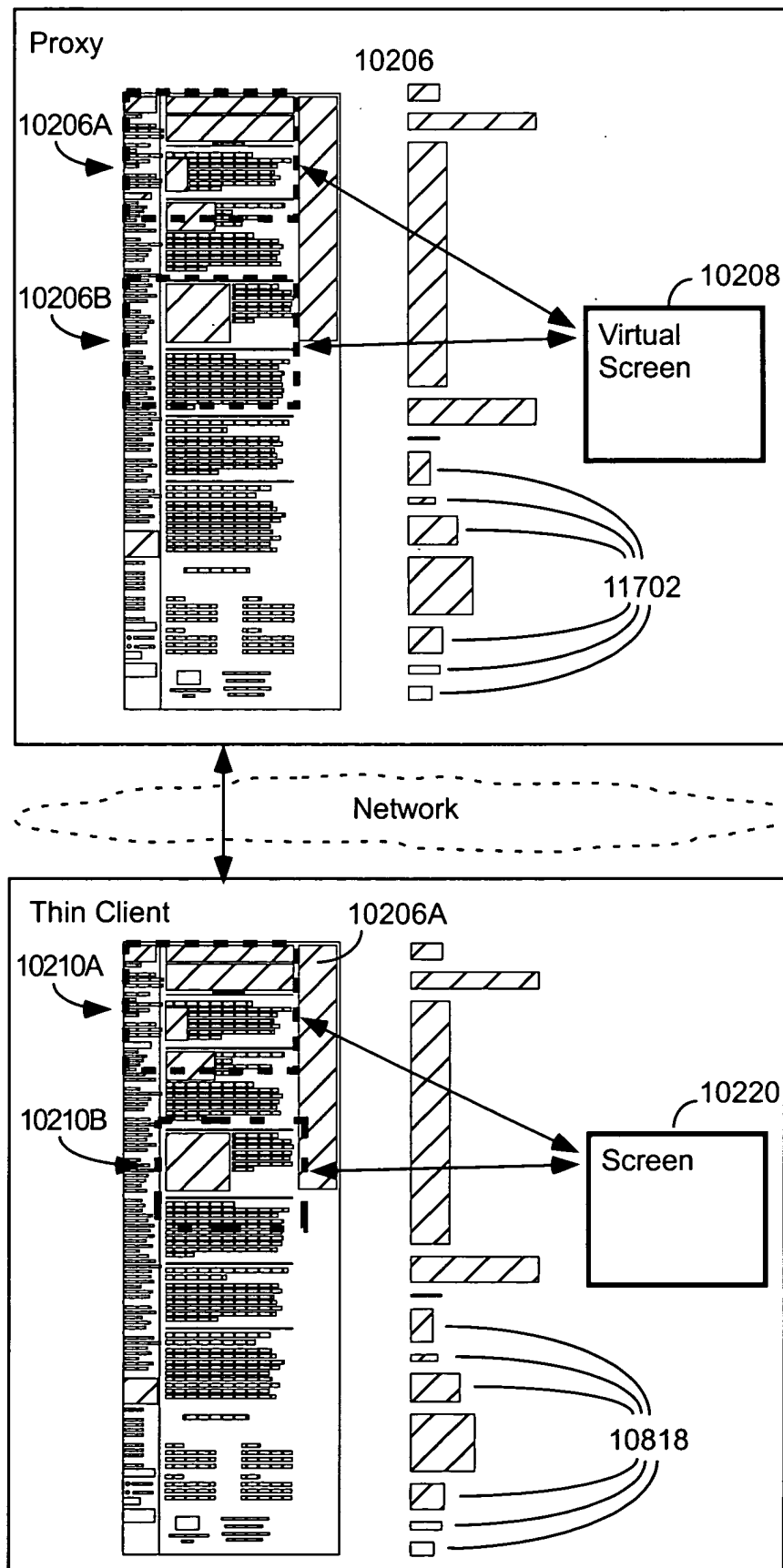
FIG. 114

- proxy browser code with use of page layout caching~11500
- ...
- if receive request for web page with an associated view setting~11502
  - request web page from server~11504
- when a web page is received from a server~11506
  - have layout engine layout web page at the virtual screen resolution specified in the view setting, substituting fonts for layout engine's measure string calls according to view setting's appropriate scale factor~11507
  - select virtual screen position relative to resulting layout which will fit the view window implicit in the view setting, and redraw screen~11508
- when receive image referenced in web page~11518
  - scale and subpixel optimize image according to scale factor~11520
- once have all images referenced in web page~11522
  - compress layout and images~11523
  - download layout followed by images~11524
- if receive request from thin client to rescale and subpixel optimize bitmaps of images at a given scale~11526
  - rescale and subpixel optimize them~11528
  - compress them~11530
  - download them to thin client~11532
- if input event from thin client~11534
  - if its layout coordinate associated with it is not currently on virtual screen~11536
    - scroll virtual screen so that it is~11538
  - calculate virtual screen coordinate corresponding to layout coordinate~11540
  - place input event with calculated virtual screen coordinate in browser's event queue ~11542
- ...

**FIG. 115**

- thin client code with use of page layout caching~11600
  - ...
  - if start receiving downloaded page layout display list~11602
    - set mapping of view window to page layout and calculate scale factor as function of view selection~11604
    - display received elements of display list which fall within view window at current mapping of screen window to display list, including current scale factor~11606
  - if user generates input to changes size or location of view window relative to downloaded layout~11616
    - make the corresponding change to the mapping of view window to display list and calculate scale factor as function any such changes~11618
    - display any portions of page layout which fall within current view window at current scale factor including~11620
      - displaying strings with font sizes which are function of current scale factor, and adjusting for disproportionate changes in size of individual or all characters as font sizes change, by changing spacing between characters~11622
    - if there is a new scale factor~11624
      - requesting proxy to re-scale at new scale factor and subpixel optimize all images on screen at new mapping~11626
      - scaling at new scale factor and re-subpixel optimizing, and displaying the previously downloaded bitmaps of all on-screen images~11628
      - when newly scaled images are received, displaying them in place of locally rescaled versions~11630
  - if user generates input event relative to screen~11632
    - use view-window-to-page-layout mapping to determine location of input in layout coordinates~11634
    - send input event and page layout coordinates to proxy~11636

**FIG. 116**



**FIG. 117**

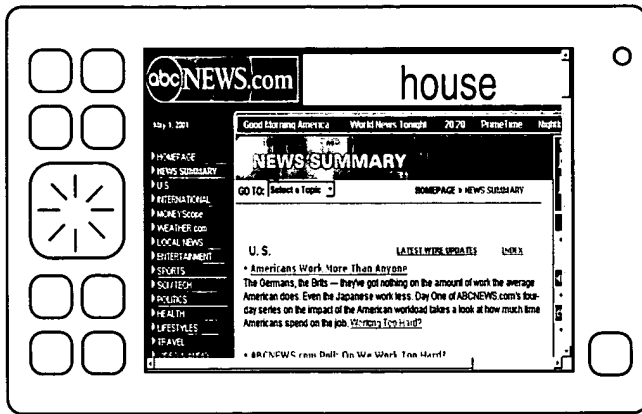
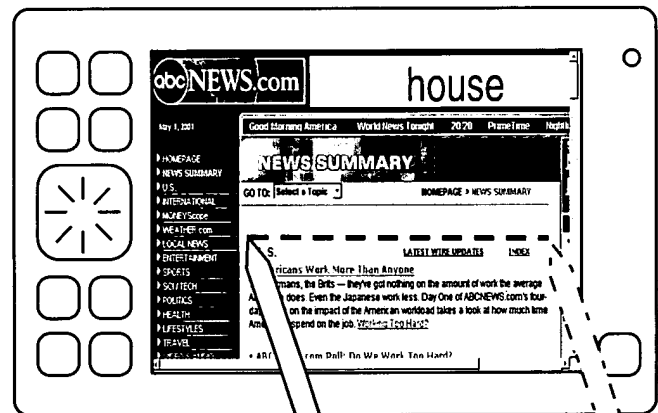


FIG. 118



11902

FIG. 119

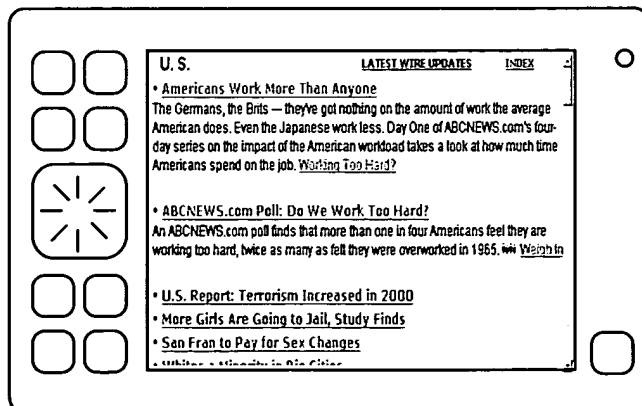


FIG. 120



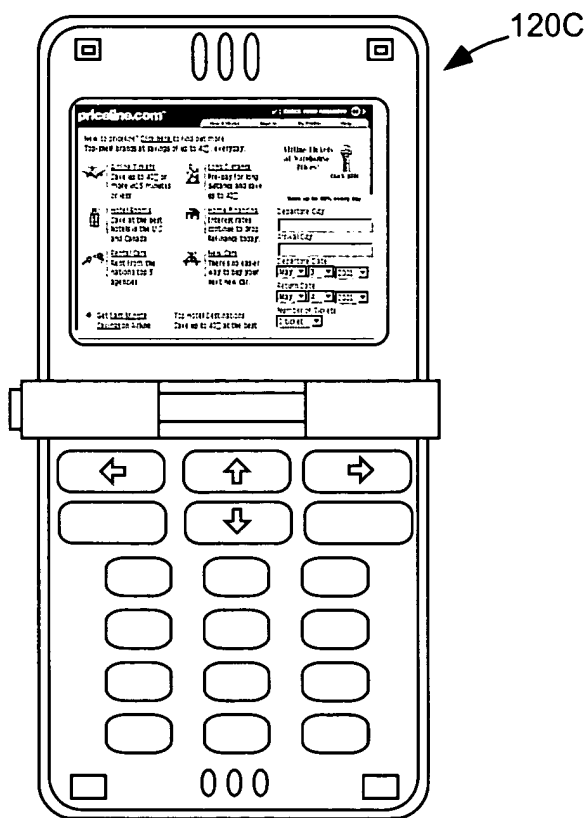


FIG. 121

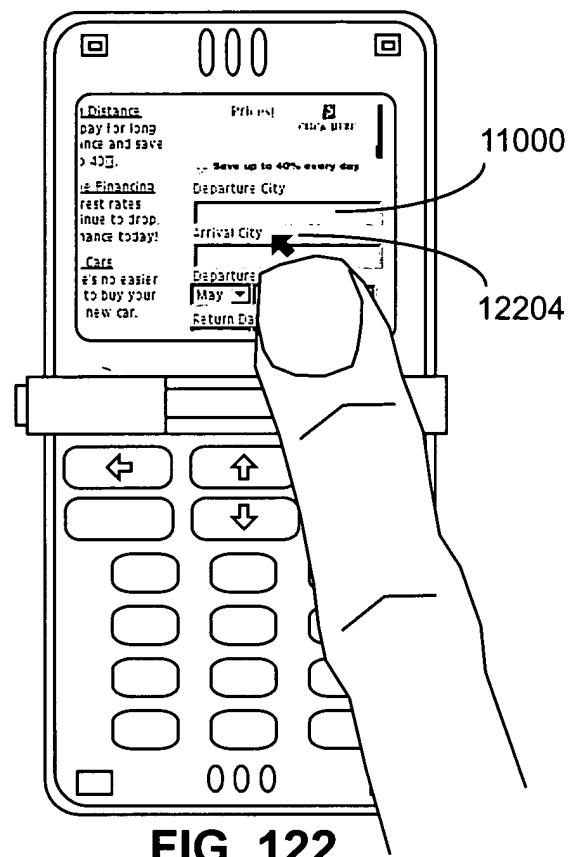


FIG. 122

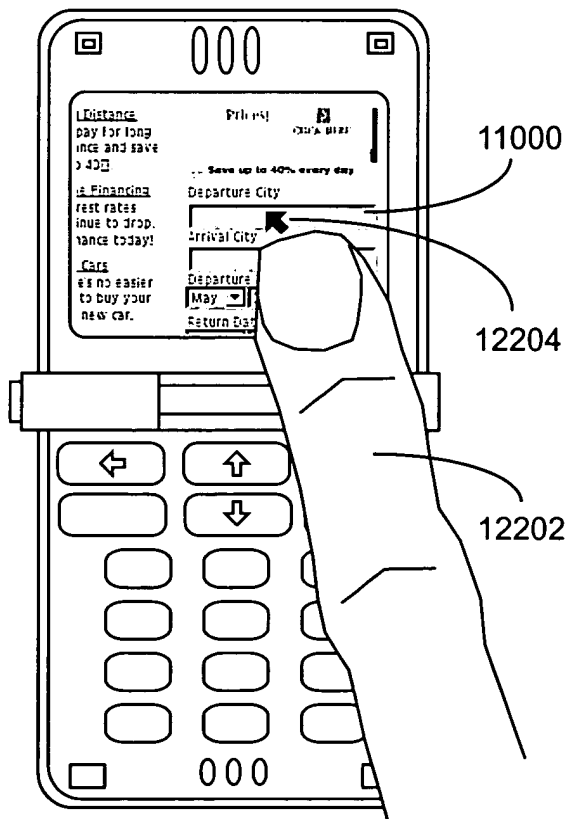


FIG. 123

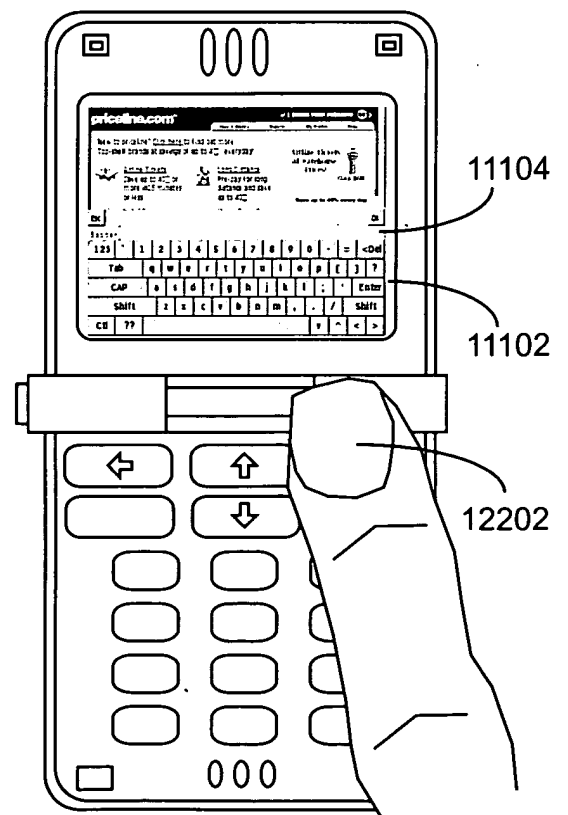
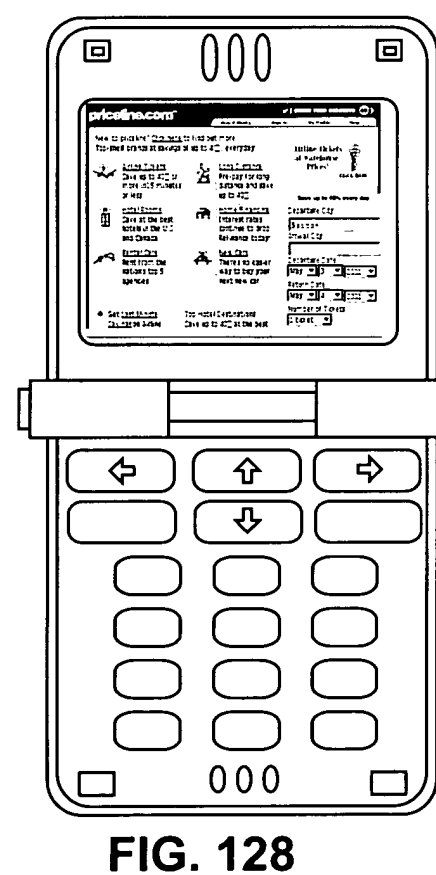
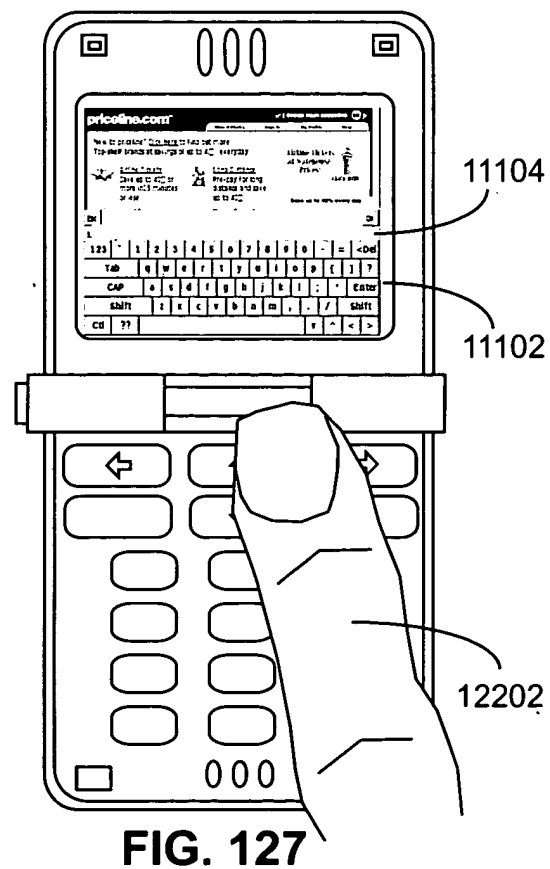
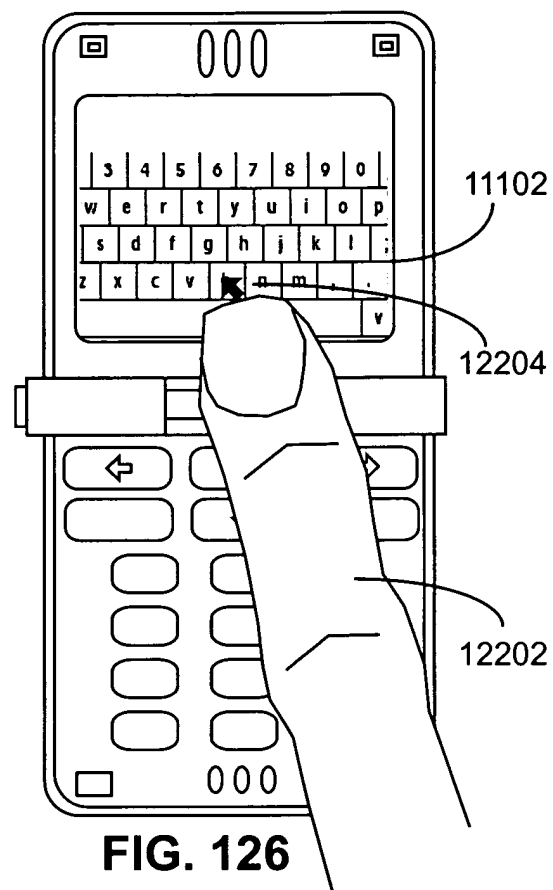
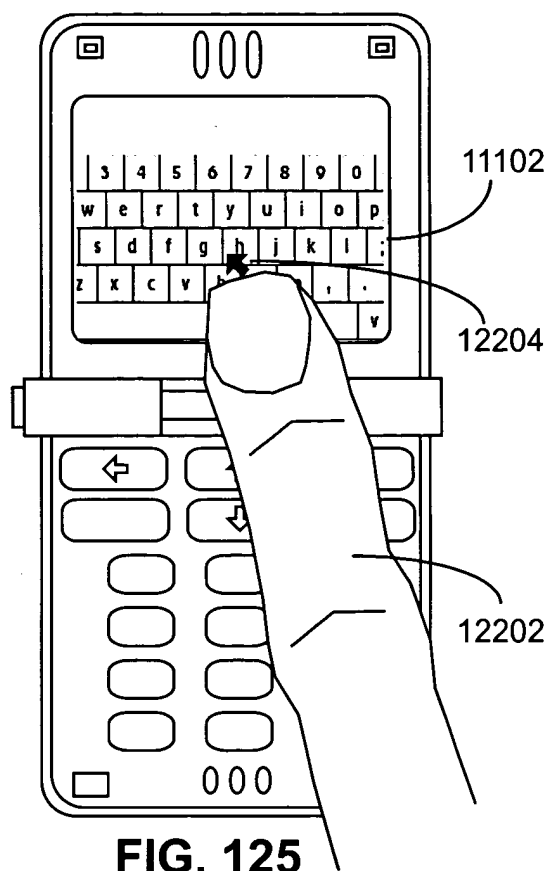


FIG. 124



- Client code for selected text reflow~12900
  - If user selects area of downloaded page layout for text reflow at a new scale factor~12902
- select all strings and corresponding underlines in layout which are substantially within selected layout area~12904
  - label any group of one or more strings whose closeness in layout indicates that they are part of the same paragraph~12906
  - reflow and display the text of each group of strings labeled as a paragraph across screen area's across boundaries at new scale factor, underlining text in reflowed lay that was underlined before~12908

## **FIG. 129**

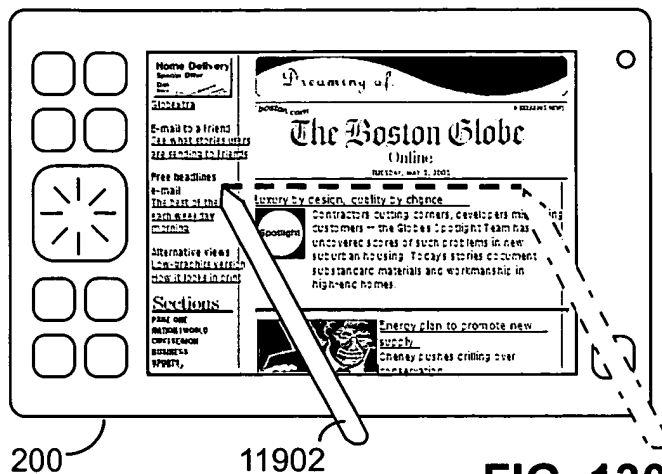


FIG. 130

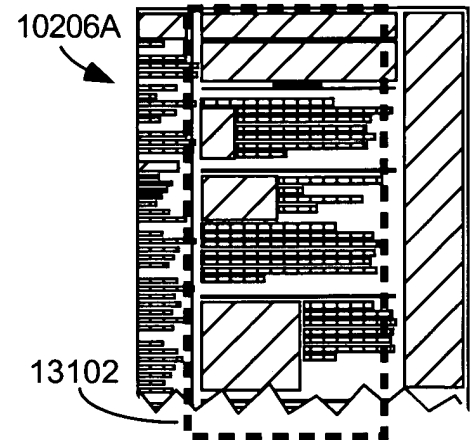


FIG. 131

Luxury by design, quality by chance

Contractors cutting corners, developers misleading customers – The Globe's Spotlight Team has uncovered scores of such problems in new suburban housing. Today's stories document substandard materials and workmanship in high-end homes

Energy plan to promote new

Supply

Cheney pushes drilling over conservation

...

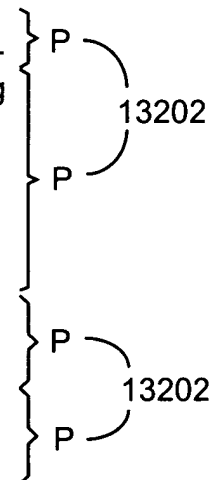


FIG. 132

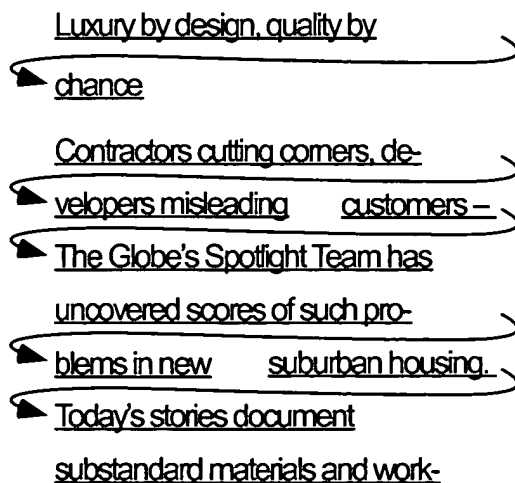
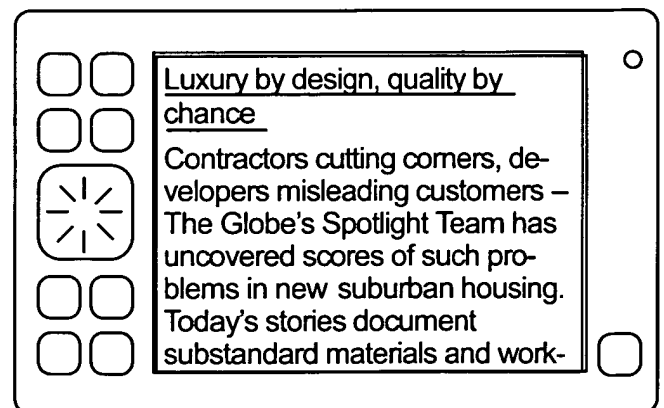


FIG. 133



200

FIG. 134

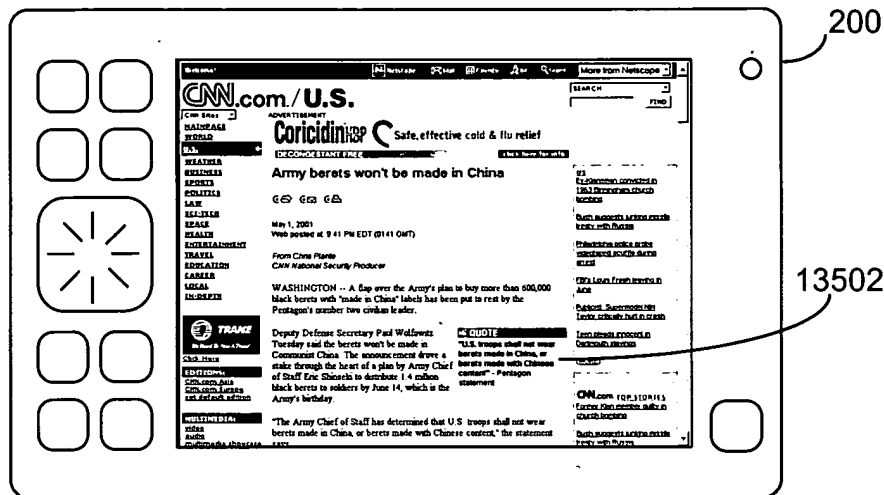


FIG. 135

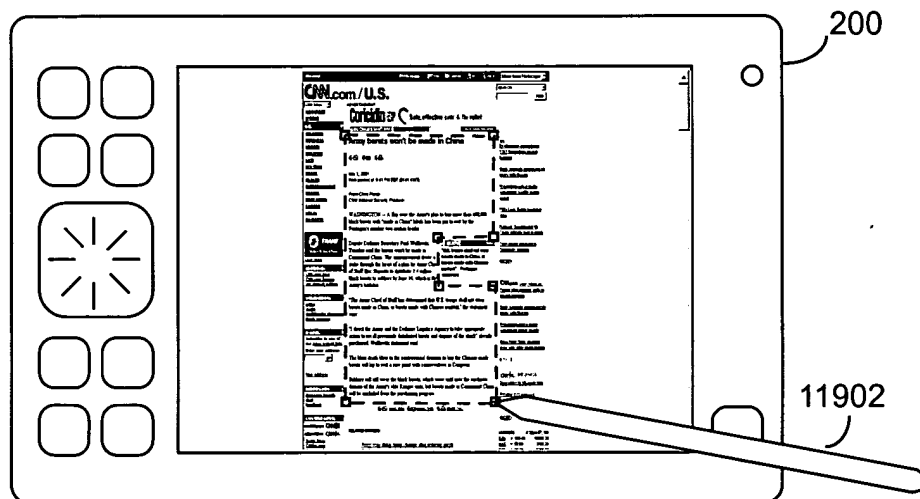


FIG. 136

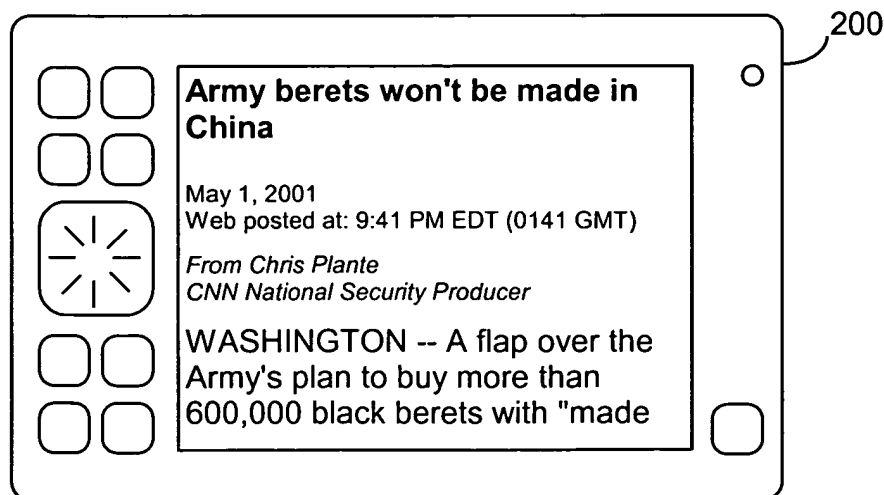
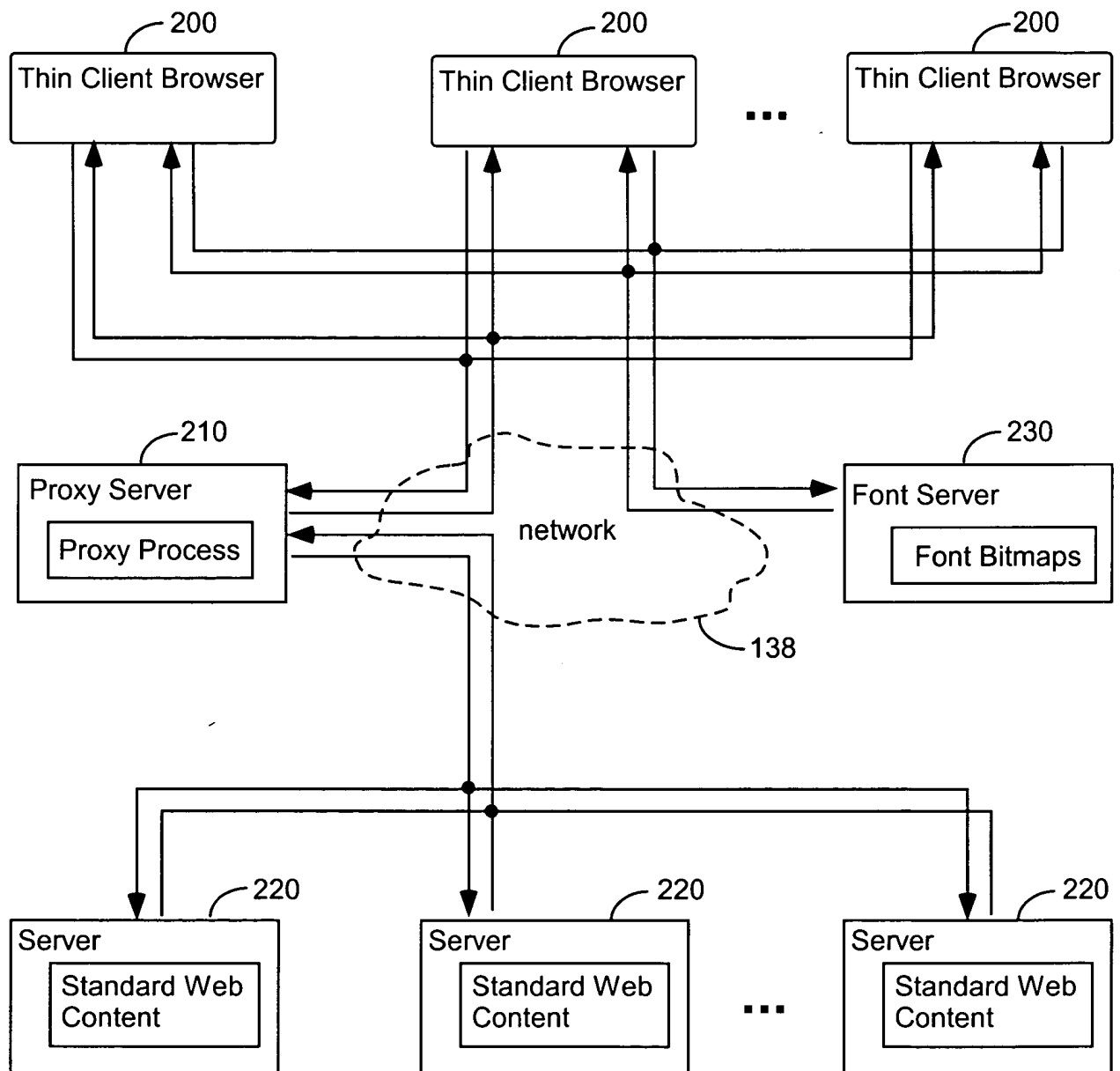


FIG. 137



**FIG. 138**

- font server code~13900
  - if receive http request for one or more characters of a font~13902
    - if there is a font file matching having path name specified in http request~13904
      - send that file over network in an http response to network address from which the font request came~13906
      - charge account associated with transaction~13908
    - else if font request is for a font bitmap~13910
      - generate font bitmap file having attributes indicated for font by path name, including~13912
        - if font request specifies that a subpixel optimized version of the font is desired~13914
          - generates subpixel optimized font of character using non-linear color balancing~13916
  - ...
  - send that file over network in an http response to the requesting address~13918
  - cache the font bitmap file at address corresponding to path name specified in request~13920
  - charge account associated with transaction~13922

**FIG. 139**

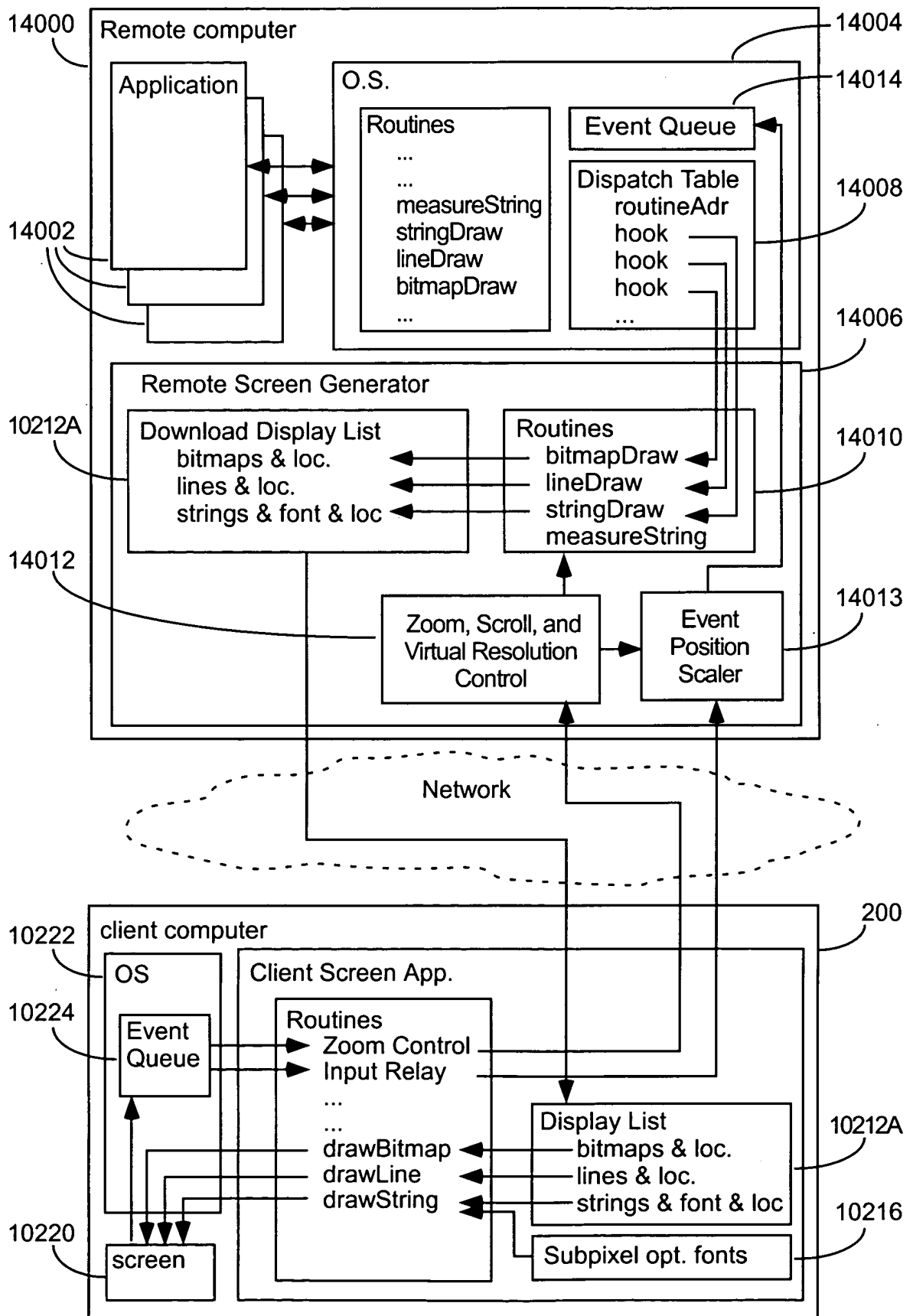
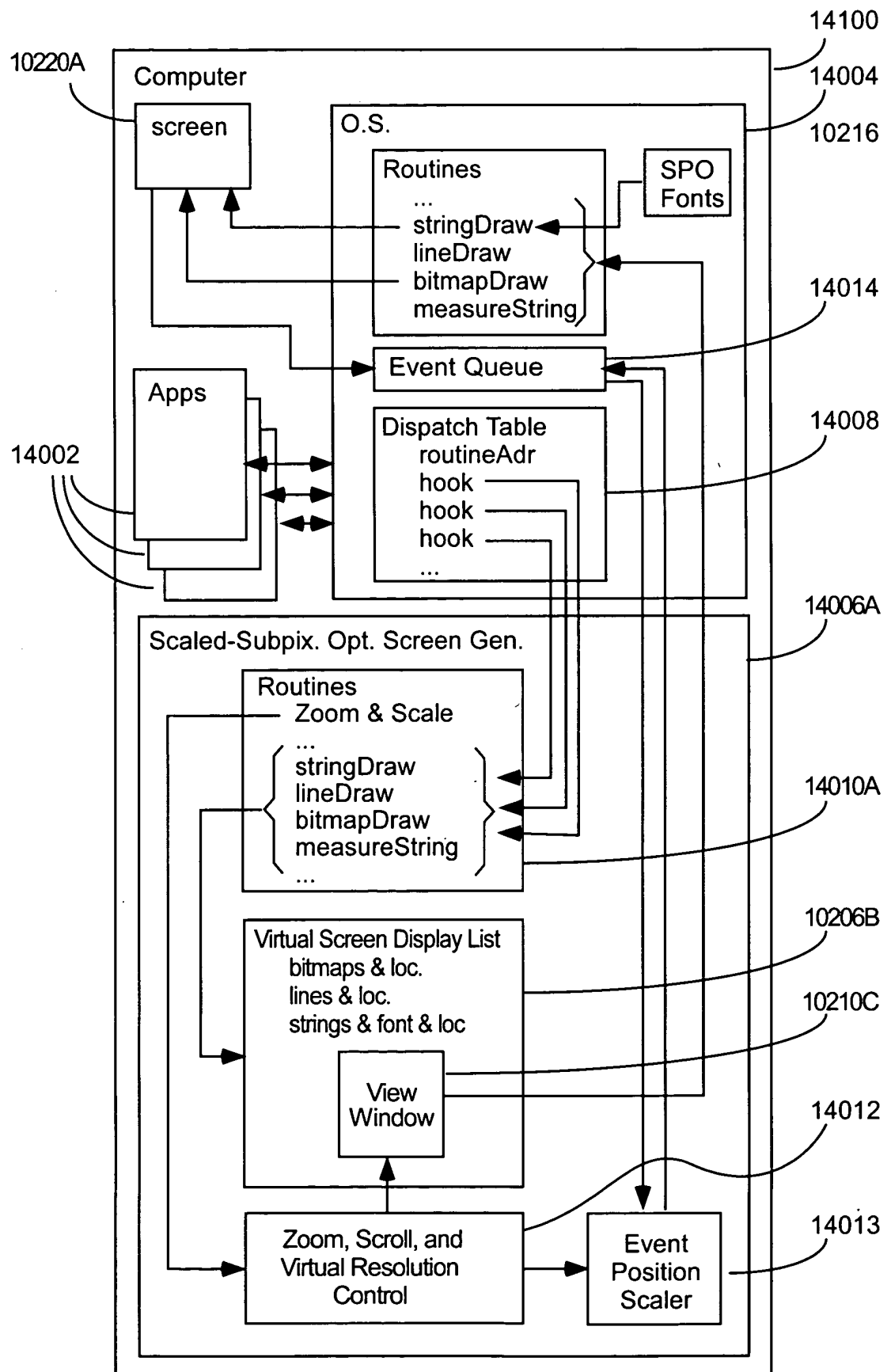


FIG. 140





**FIG. 141**

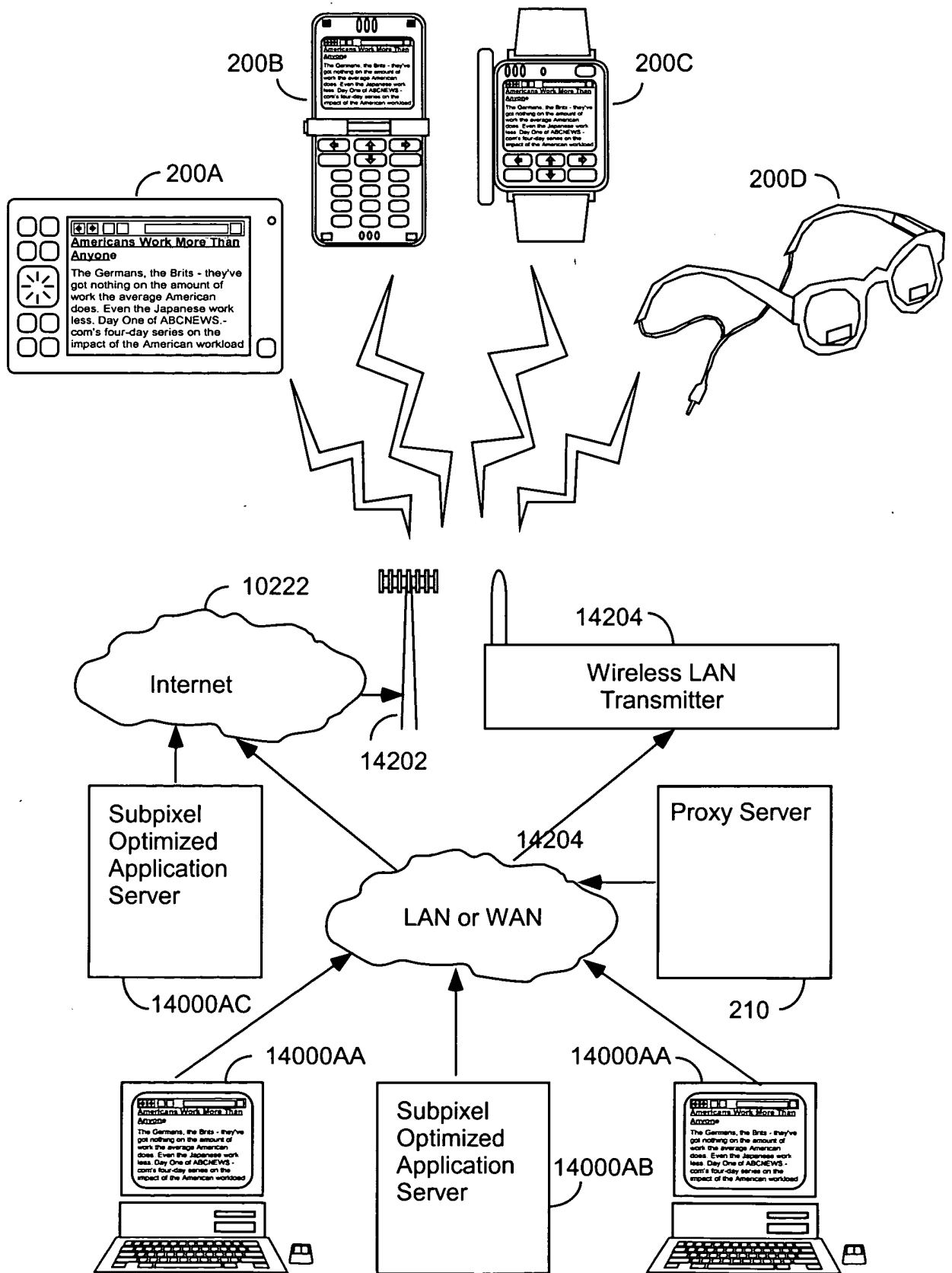


FIG. 142

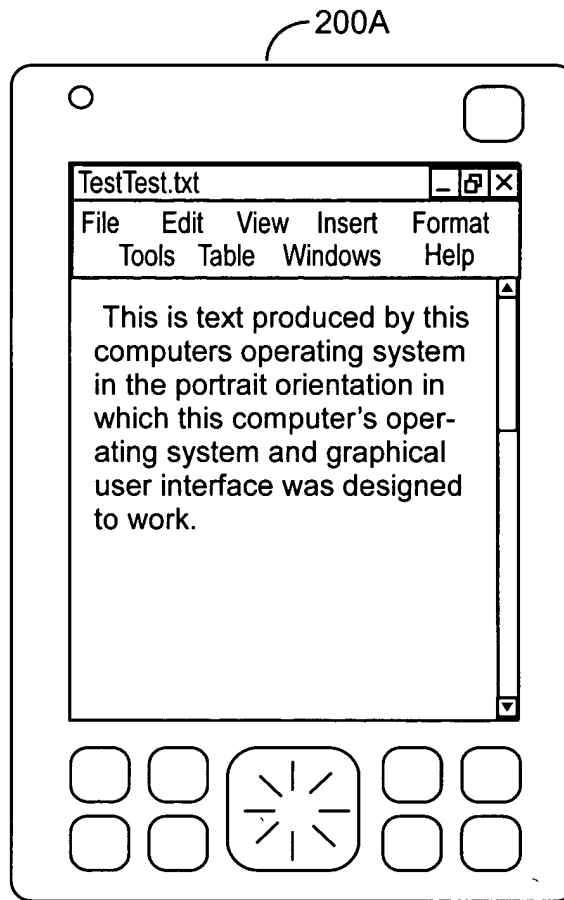


FIG. 143

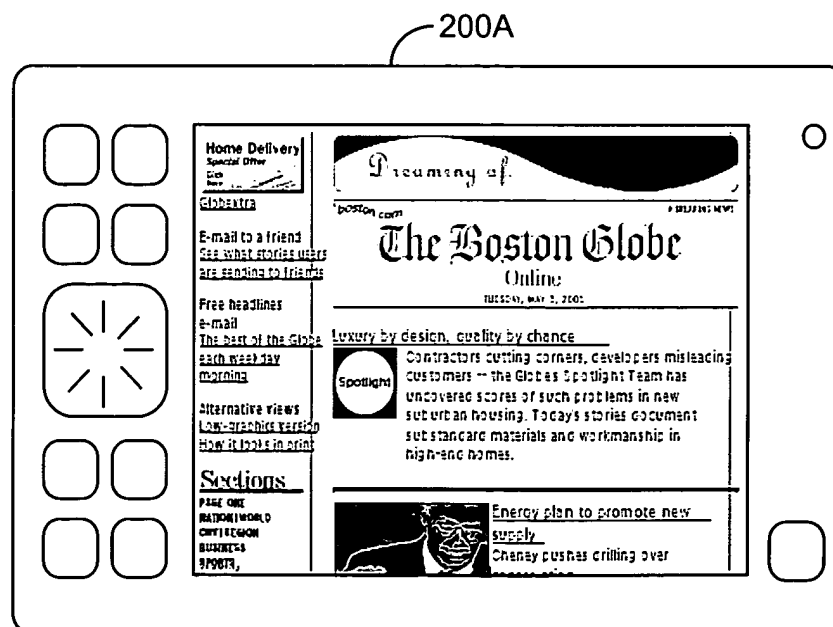


FIG. 144

- rectangleCmd~14500
  - draw rectangle with screen position, height, and width defined with higher resolution than screen pixel resolution, using bicolor subpixel optimization and using current background color~14502

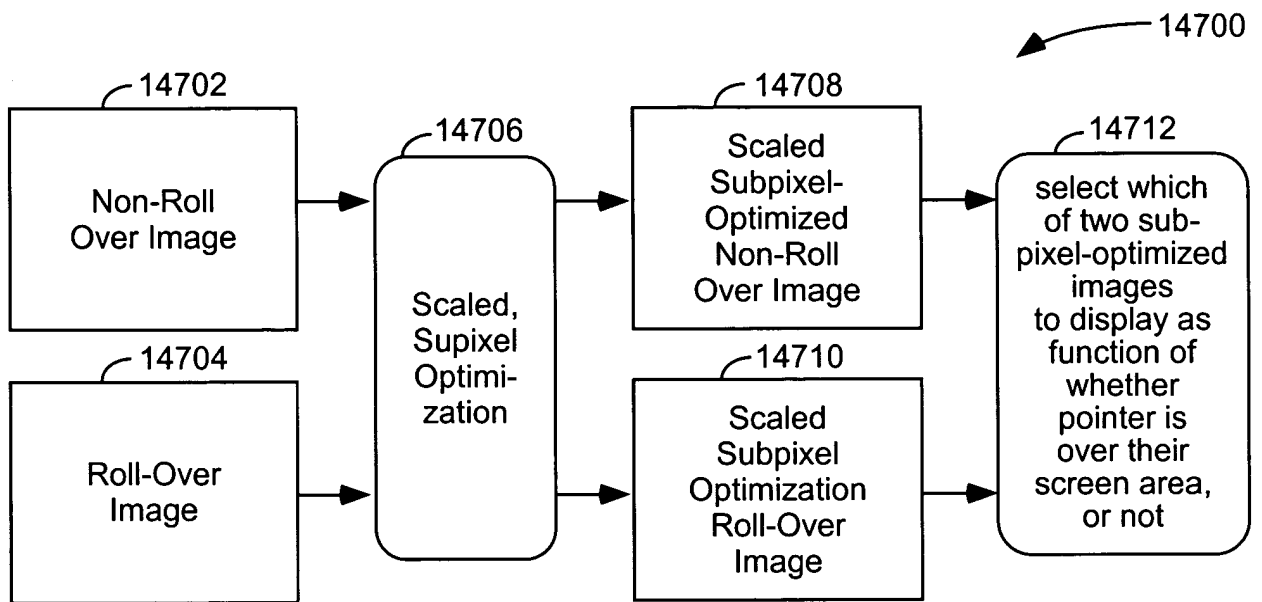
## FIG. 145

- downloading web applets which creates subpixelized elements on screen~14600
  - server and/or proxy~14602
    - in response to request for media from thin client download media including applets~14608
  - client~14604
    - request media~14606
    - receive media including applet~14610
    - load and run applet~14612
    - applets draw subpixel optimized elements to subpixel addressable screen on client~14614

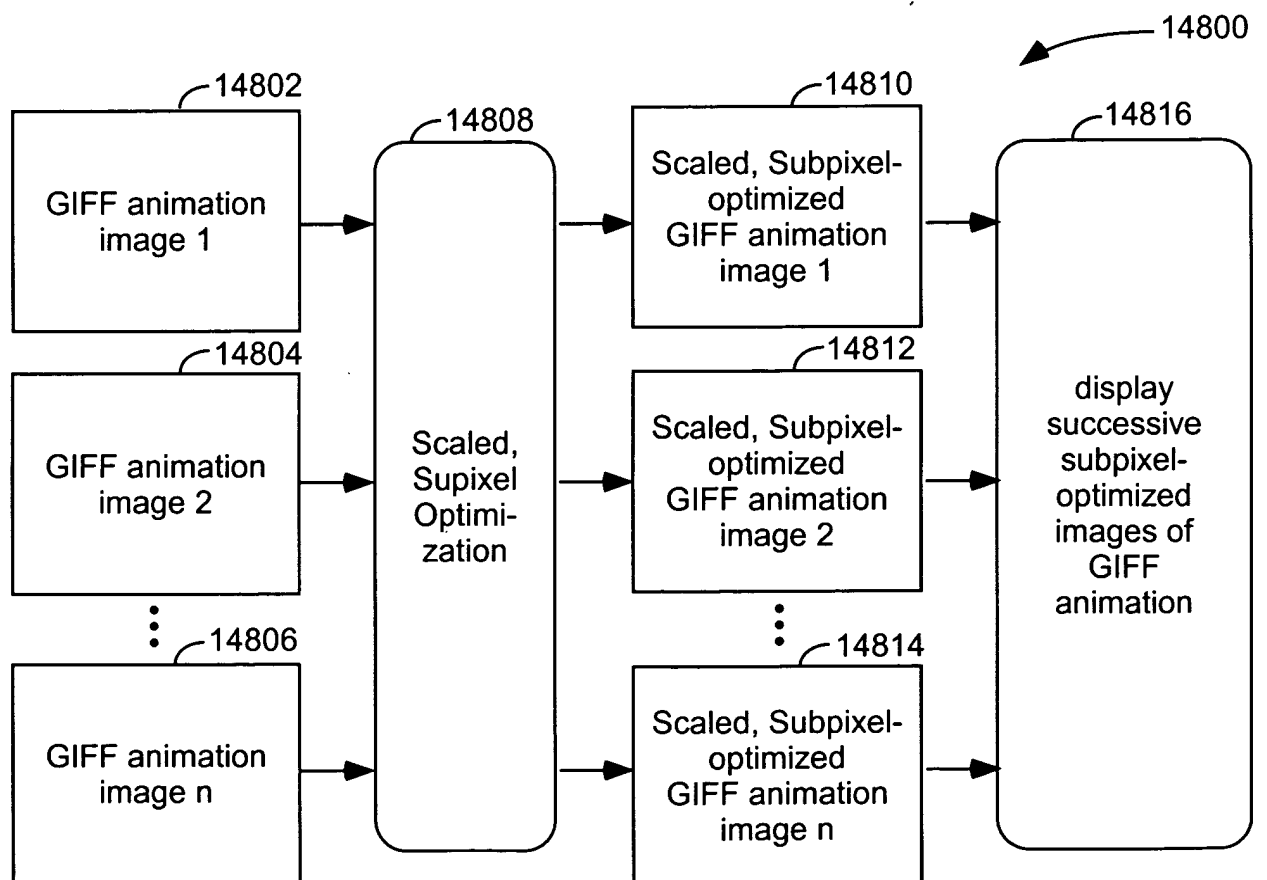
## FIG. 146

- subpixel optimization of 3-d animations~14900
  - for each of successive frame times~14902
    - run 3-d animation engine to create bitmap of current frame, or at least of those portions of image which have changed since the last frame, at a higher resolution than the resolution at which subpixel-optimized images will be displayed~14904
    - scale-down and subpixel optimize frame bitmap, or at least of changes in it since the last frame~14906
    - display scaled-down subpixel optimized frame bitmap, or at least scaled subpixel optimized bitmaps of changed portion of frame~14908

## FIG. 149



**FIG. 147**



**FIG. 148**

-game server computer~15000

- if have receive user input from one or more game client computers~15002
  - feed it to game engine~15004

-...

- have game engine compute display list for current frame (or changes to display list for current frame)~15006

- have 3-d rendering routine render bitmap frame of current display list(or current changes to display list) at higher resolution than that at which corresponding subpixel-optimized bitmaps will be shown~15008

- scale-down and subpixel optimizie current frame bitmap(or bitmaps of current changes to frame and their scaled down locations)~15010

-...

- compress one or more successive scaled-down subpixel optimized bitmaps (and their locations) ~15012

- download compressed, scaled, subpixel-optimized animation frames (or changes and their locations) to game client~15014

## **FIG. 150**

-game client~15100

-...

- receive downloaded images (and screen locations)~15101

- decompress animated images (and screen locations)~15102

- displays the scaled, subpixel optimized animation frame bitmaps (or change bitmaps at their respective positions)~15104

- if have received any user input~15106

- upload user input to game server~15108

-...

## **FIG. 151**

- subpixel optimization of images with transparency maps~15200
  - produce scaled, either a bicolor or multicolor subpixel-optimized bitmap of the foreground image~15202
  - produce a correspondingly scaled, bicolor subpixel optimized bitmap of the images transparency map~15204
  - display foreground image's bitmap on a subpixel optimized display including:~15206
    - for each pixel row of the displayed image~15208
      - for each subpixel of such row~15210
        - set currentAlpha to the alpha value of the corresponding subpixel of the transparency map~15212
        - set the luminosity of the current subpixel to currentAlpha times the luminosity of the corresponding subpixel of the foreground image plus  $(1 - \text{currentAlpha})$  times the prior luminosity value of the current subpixel~15214

## FIG. 152

- subpixel optimizing video having interpolation between keyframes~15300
  - decompress video~15302
  - scale and subpixel optimizing key frames~153040
  - scale but do not subpixel optimize interpolated changes between keyframes because of its rapid speed~15306
  - display scaled video on subpixel addressable display with subpixel optimized keyframes and non-subpixel optimized interframe interpolation~15308

## FIG. 153

- subpixel optimizing video representing changes to portions of frame~15400
  - decompress video~15402
  - scale and subpixel optimizing frames~15404
  - scale and subpixel optimize change bitmaps, scale their location relative to frame~15406
  - repeatedly display on subpixel addressable display~15407
    - any scaled, subpixel optimized video frame followed by a sequence of one or more scaled subpixel optimized change bitmaps over the bitmap of that frame at corresponding scaled positions on the frame~15408

## FIG. 154

- moving images with fixed subpixelation~15500
  - store subpixel-optimized bitmap of image~15502
  - for each successive frame time~15503
    - calculate movement of image at fixed size and orientation, rounding location to nearest horizontal and vertical whole pixel location~15504
    - display image at that location~15506

## **FIG. 155**

- moving image with changing subpixelation~15600
  - store high resolution source image~15602
  - for each successive frame time~15603
    - calculate translation, rotation, and/or transformation of high resolution source~15604
    - generate scale-down and subpixel optimized bitmap of image with images mapping into subpixel grid associated with bitmap being a function of its translation, rotation, and/or transformation~15606
    - display resulting subpixel optimized bitmap on subpixel display~15608

## **FIG. 156**

- subpixel optimization of DVD video~15700
  - decompress DVD video to a resolution higher than that at which it is to be displayed in subpixel optimized image~15702
  - scale and subpixel optimize decompressed bitmaps of video images~15704
  - display scaled subpixel optimized bitmaps of video images on subpixel addressable display~15706

## **FIG. 157**

- subpixel optimization of HDTV~15800
  - decompress HDTV video to a resolution higher than that at which it is to be displayed in subpixel optimized image~15802
  - scale and subpixel optimize decompressed bitmaps of video images~15804
  - display scaled subpixel optimized bitmaps of video images on subpixel addressable display~15806

## **FIG. 158**



- subpixel optimization of mpeg4~15900
  - receive and decompress mpeg4 video~15902
  - use bicolor subpixel optimization with non-linear color balance in scaling down of bicolor objects~15904
  - use multi-color subpixel optimization in scaling down of non-bicolor objects~15906
  - display combination of bicolor and multicolor objects on subpixel optimized display moving subpixel optimized objects relative to screen~15908

## **FIG. 159**

- server subpixel optimization of scaled down, downloaded video~16000
  - receives request for video and specification of subpixel display resolution~16002
  - receives requested video content~16004
  - scales down and subpixel optimizes the received video to subpixel resolution association with request~16006
  - compresses video~16008
  - downloads it to requesting device~16010

## **FIG. 160**

- proxy subpixel optimization of scaled down, downloaded video~16100
  - proxy computer code~16100
    - when receive request for video (and specification of subpixel resolution)~16102
      - send corresponding request for the requested video to a server~16103
    - when receive requested video content~16104
      - scale down and subpixel optimizes video (to specified subpixel resolution)~16106
      - compress subpixel optimized video~16108
      - download it to thin client~16110
  - thin client code~16112
    - in response to user input, send request for video to proxy (including subpixel resolution at which video is to be displayed)~16113
    - when receive requested video from proxy~16114
      - decompress video~16115
      - display scaled-down decompressed video on subpixel addressable display~16116

## **FIG. 161**

- Electronic ink code~16200
  - if user enters electronic ink input~16202
    - record strokes as series of points and curves or lines in between~16204
    - draw ink on screen using subpixel optimization of lines and curves with non-linear color balance~16206
  - if user selects to scale up representation of electronic ink~16208
    - produce subpixel optimized bitmap of ink's lines and curves using bicolor subpixel optimization with non-linear color balancing at the selected scaled up size~16210
    - display scaled up image~16212
  - if user selects to scale down representation of electronic ink~16214
    - to produce subpixel optimized bitmap of ink's lines and curves using bicolor subpixel optimization with non-linear color balancing at the selected scaled down size~16216
    - display scaled down image~16218

**FIG. 162**

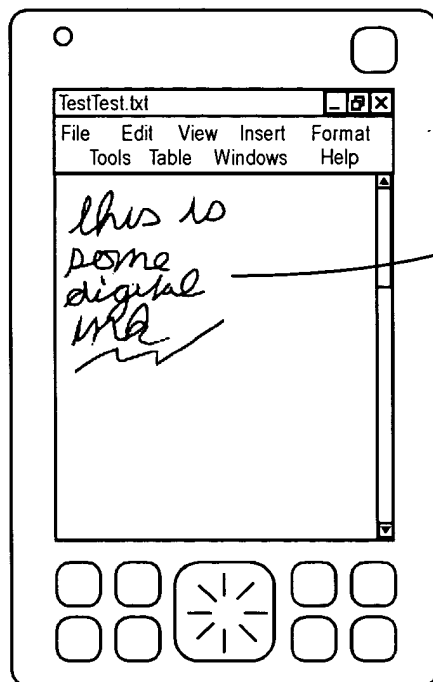


FIG. 163

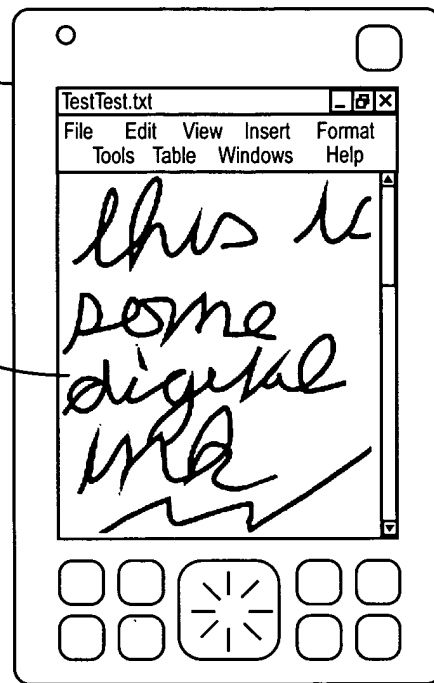


FIG. 164

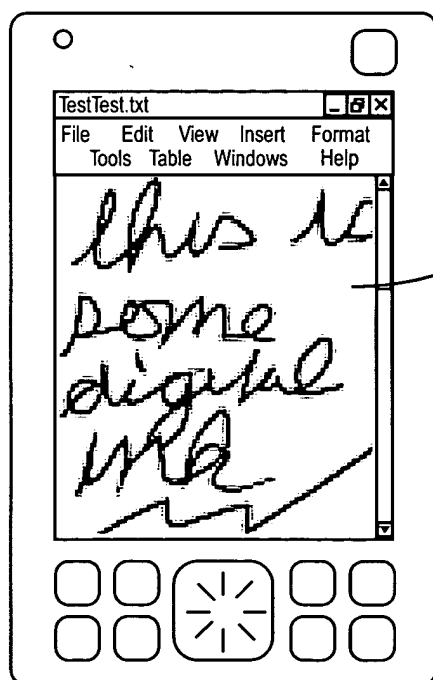


FIG. 165

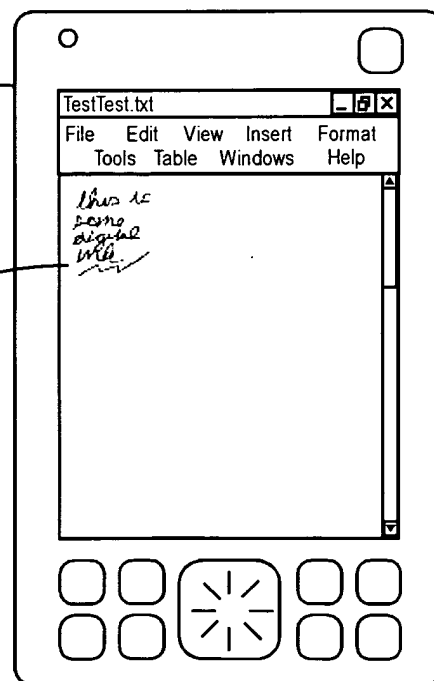
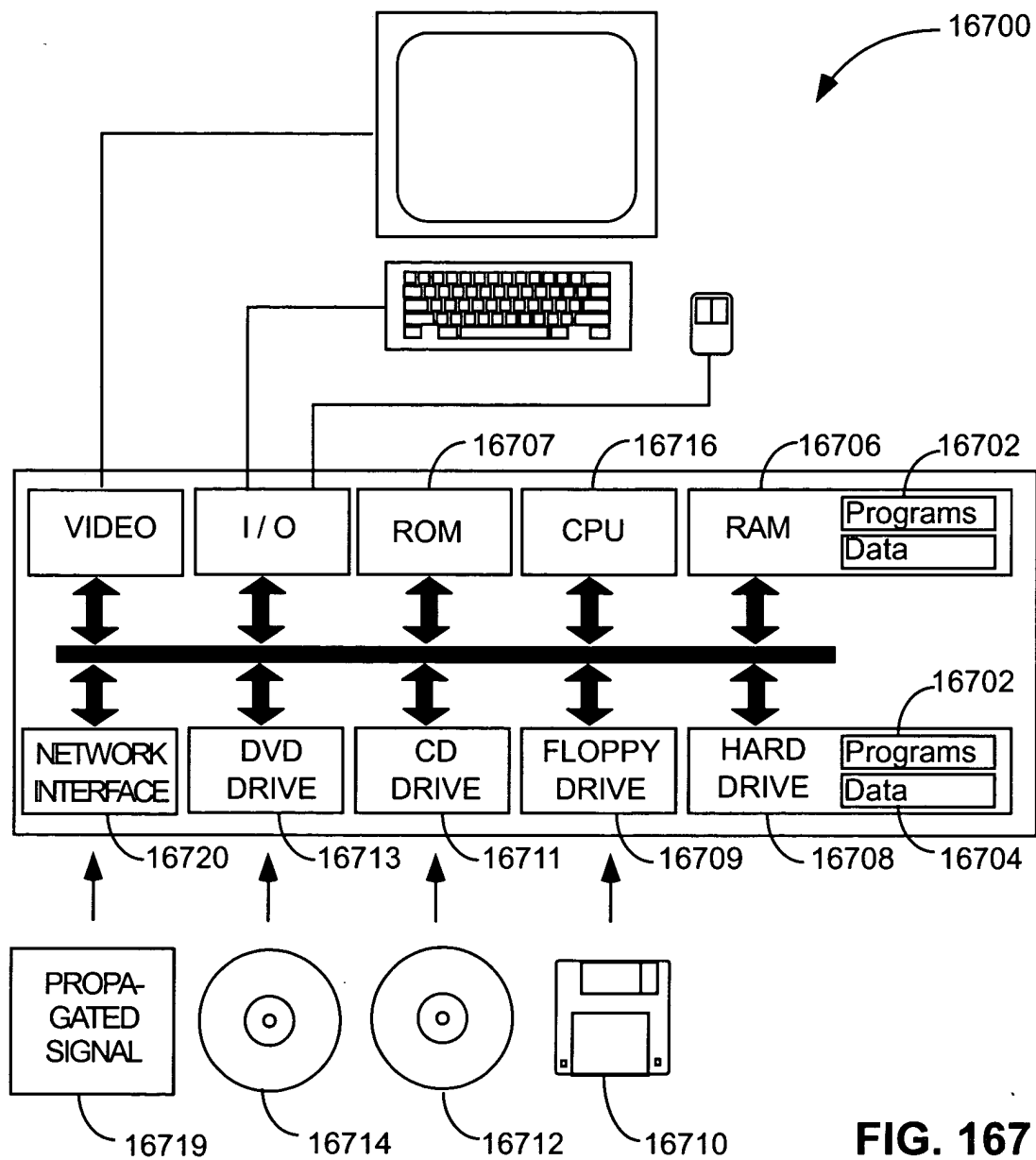


FIG. 166



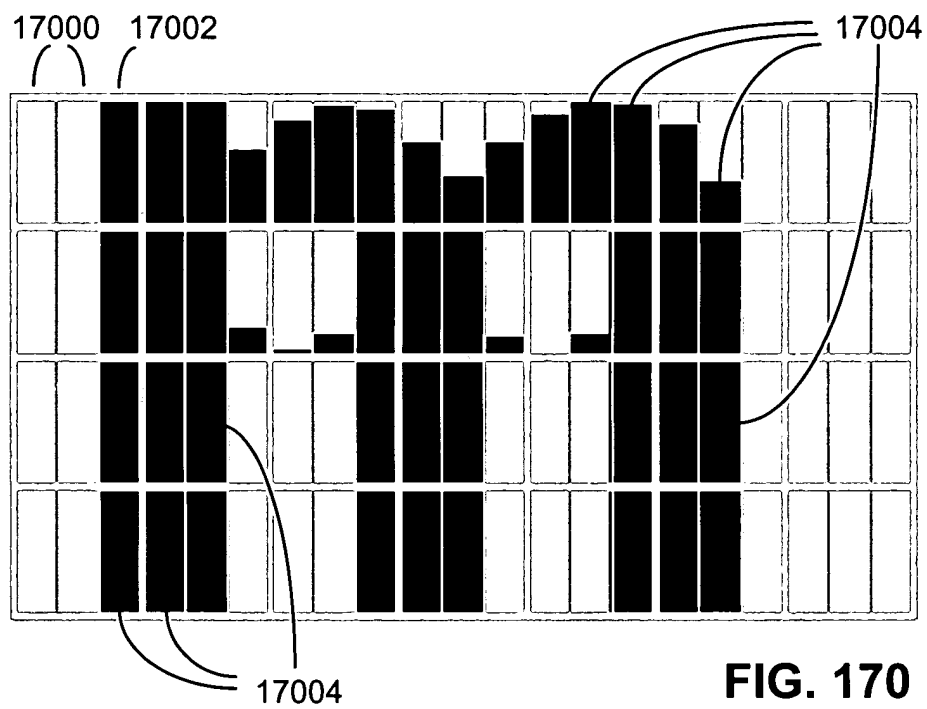
**FIG. 167**



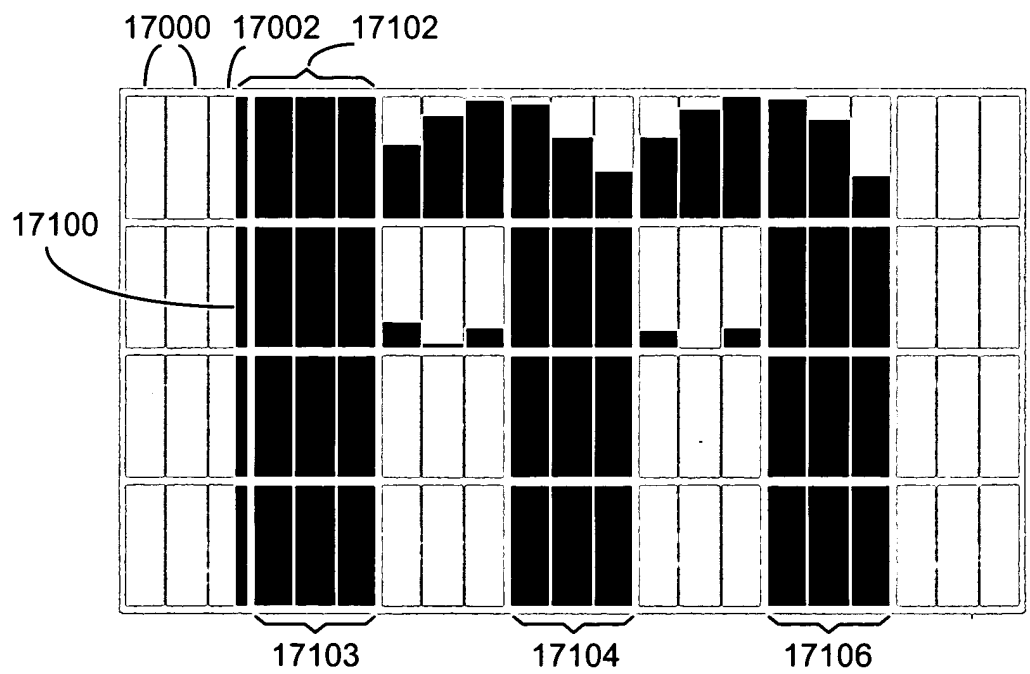
FIG. 168

The global leader in  
search infrastructure.  
LookSmart provides  
quality search

FIG. 169



**FIG. 170**



**FIG. 171**

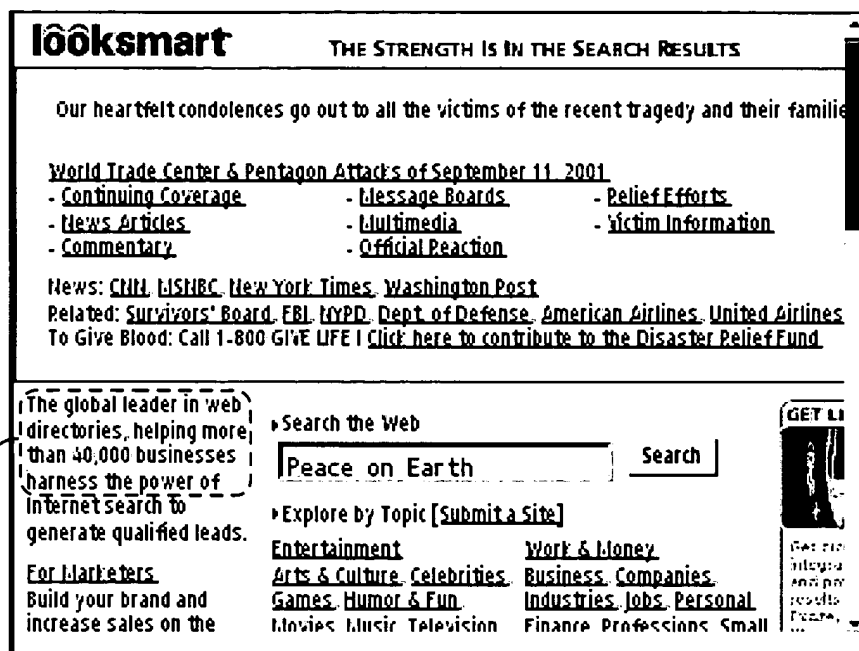


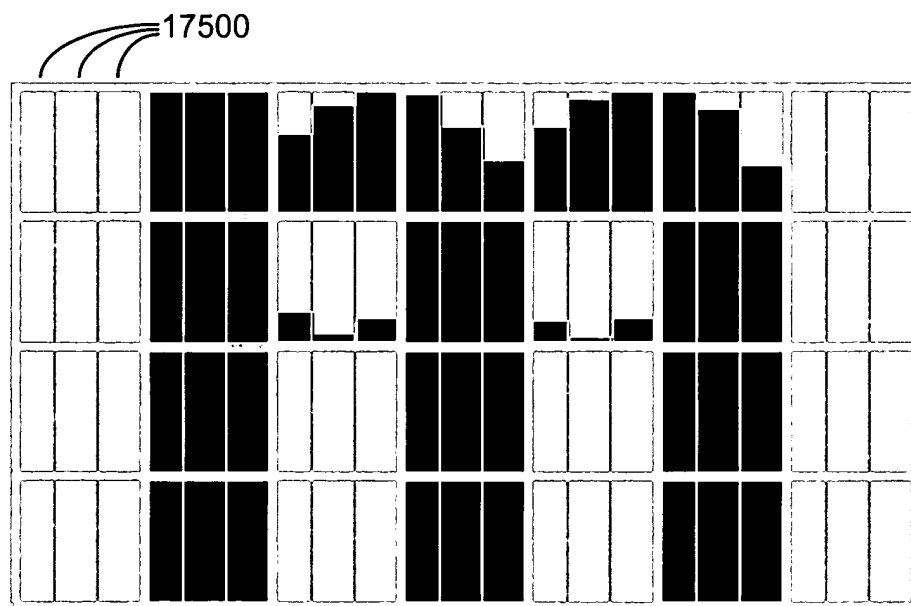
FIG. 172

The global leader in web directories, helping more than 40,000 businesses harness the power of

FIG. 173

ing more

FIG. 174



**FIG. 175**



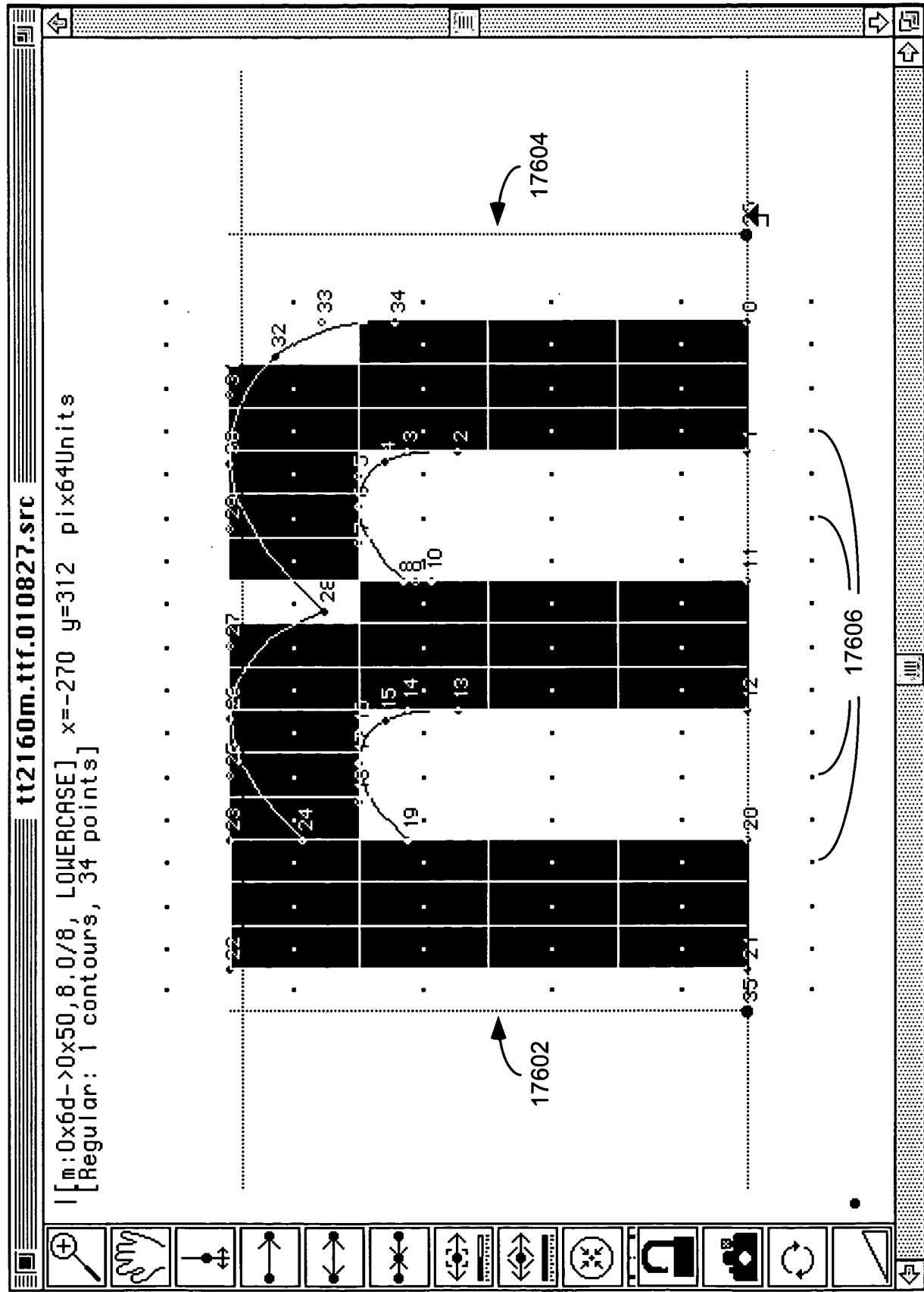


FIG. 176

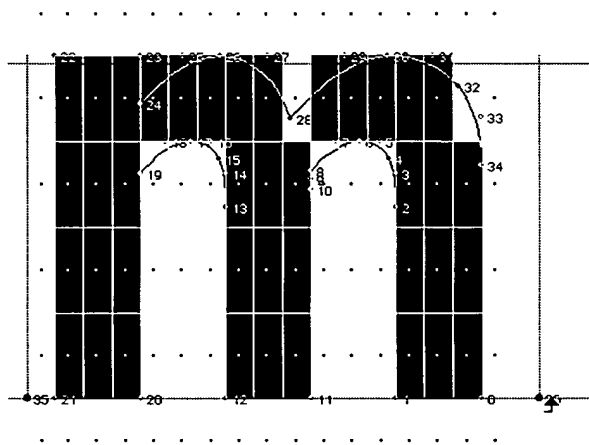


FIG. 177

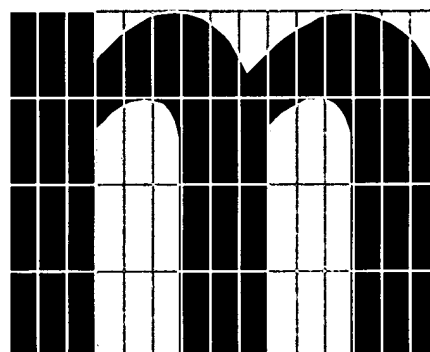


FIG. 178

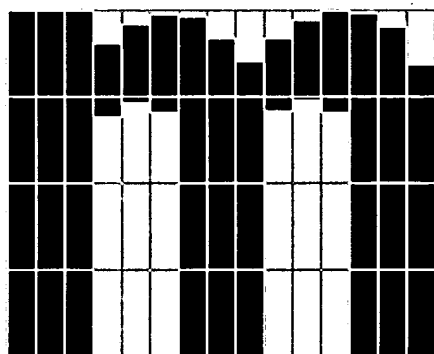


FIG. 179

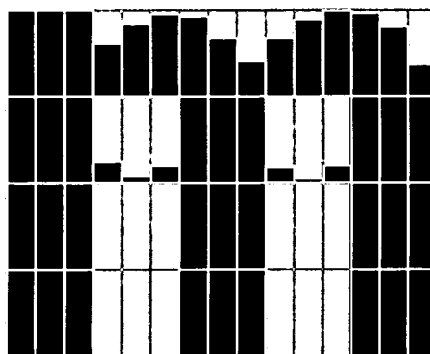


FIG. 180

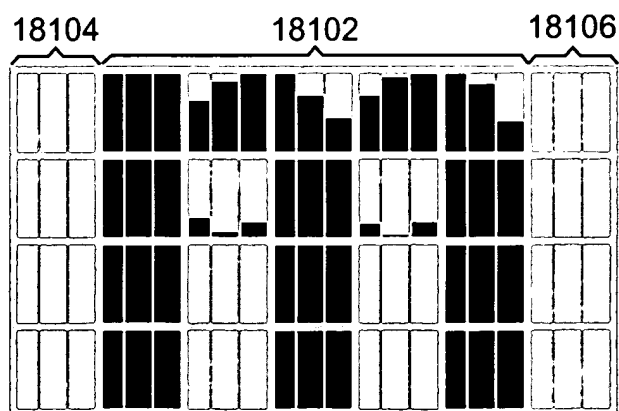


FIG. 181

- subpixel optimized font bitmaps with non-linear color balance~6000A
  - determine tightest rectangular array of rasterization units into which character font shape fits, taking into account alignment of character-font shape relative to rasterization units due to hinting18202
  - for each pixel row~6002A
    - for each subpixel in a row~6004
      - determine a coverage value representing the percent of the subpixel which is covered by the font shape~6006
  - map resulting array of subpixel coverage values into an array of subpixel addressable pixels, aligning first column of rasterization units with leftmost subpixel of a pixel row~18204
  - pad array with pixel column comprised of three subpixel's each to left~18206
  - pad array with two, three, or four more sub-pixel columns to right, so as to cause the total number of sub-pixel columns to be an even multiple of three~18208
  - adjust left and right side bearing values to compensate for padding subpixel column on left and right side of bitmap~18210
  - perform non-linear color balancing~18212
  - convert to packed color value pixel bitmap~18214

## FIG. 182

- drawing character string~18300
  - set pen position to start position for string~18302
  - for each character of string to be displayed~9714A
    - access its associated font bitmap~9716
    - set character start position to pen position~18304
    - adjust pen position by left side bearing~18306
    - for each pixel value of the font bitmap~9718
      - if pixel is non-zero, draw pixel18308
    - set pen position to character start position plush current character's advance width~18310

## FIG. 183

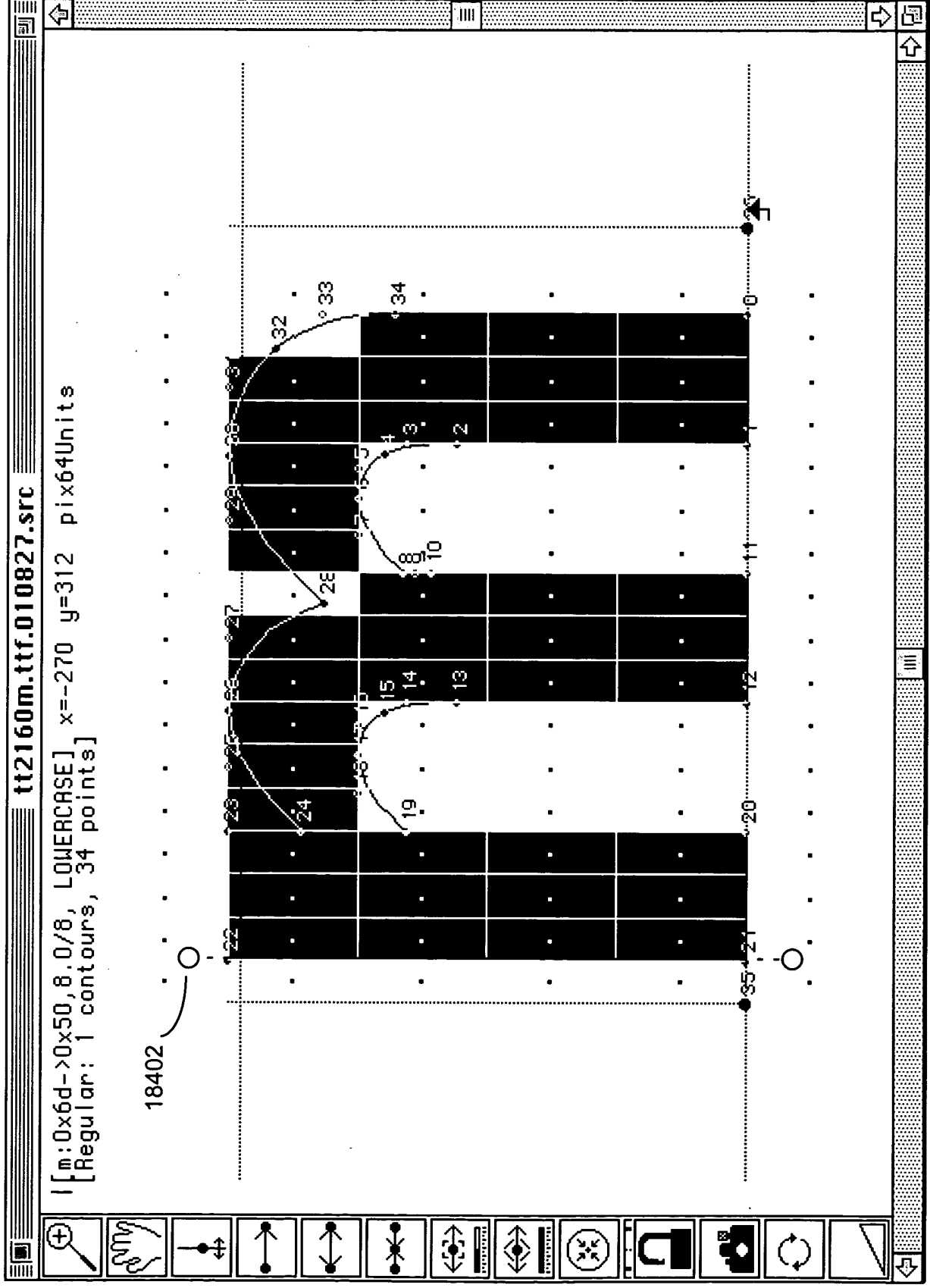


FIG. 184

- selected text reflow~18500
  - access web page's contents~18502
  - perform first layout of web page's contents, placing text at different horizontal locations as indicated by web page~18504
  - display elements of web page at positions determined by the first layout~18506
  - enable user to select a portion of text at a given horizontal location in display of first layout~18508
  - respond to user selection of such a portion of text by~18510
    - performing a second layout of the selected text by re-flowing it in a new column, at a different font size relative to new column's width~18512
    - displaying layout of new column at scale that fills at least two thirds of width of screen~18514

## FIG. 185

- zoom to fit~18600
  - access web page's contents~18602
  - perform layout of web page's contents~18604
  - display all or portion of layout at first scale~18606
  - enable user to drag pointing device across first scale layout display; ~18608
  - if drag continues across screen boundary ~18610
    - scroll onto screen portions of layout at first scale previously off screen~18612
  - if drag is released~18614
    - define selected layout part based on position in first scale layout display of start and end of drag ~18616
  - display selected part of layout at second scale that fits selected layout part to screen. ~18618

## FIG. 186

- drag scroll ~18700
  - access web page's contents~18702
  - perform layout of web page's contents~18704
  - display all or portion of layout~18706
  - enable user to drag a pointing device across display of layout~18708
  - responding to any such drag across a boundary associated with a screen edge by scrolling onto screen, past the screen edge, portions of layout previously off screen~18710

## FIG. 187

- clickzoom~18800
  - access web page's contents~18802
  - perform layout of web page's contents~18804
  - display all or portion of layout at first scale~18806
  - enable user to click a pointing device at a selected location in display of layout at first scale~18808
  - responding to such a click by performing a zoomed display of portion of layout around selected location~18810

## **FIG. 188**

- zoomclick~18900
  - access web page's contents~18902
  - perform layout of web page's contents~18904
  - display all or portion of layout at first scale on touch screen~18906
  - if user presses touch screen at first position in first scale layout display~18908
    - replace first scale layout display with a display at larger scale of portion of layout that includes first position at substantially same position on screen as in first scale layout display~18910
    - display cursor slightly above location of touch~18912
    - respond to any movement of touch by correspondingly moving cursor on display at second scale~18914
    - respond to any movement of touch across a boundary associated with a screen edge by scrolling onto screen, past the screen edge, portions of layout at second scale previously off screen~18916
  - if user subsequently releases of press at a selected position in second scale layout display~18918
    - act as if a mouse click had occurred at corresponding position relative to web page~18920
    - replace second scale layout display with first scale layout display~18922

## **FIG. 189**

- zoom out with greeking~19000
  - access web page's contents~19002
  - perform layout of web page's contents~19004
  - if user has selected given larger display scale~19006
    - display portion of web page's layout at larger scale, including~19008
      - representing layout's images with bitmap images scaled for display at larger scale~19010
      - representing layout's strings by bitmap images composed of font bitmaps sized for display at larger scale~19012
  - if user has selected given smaller display scale~19014
    - display portion of web page's layout at smaller scale, including~19016
      - representing layout's images with bitmap images scaled for display at smaller scale~19018
      - representing layout's strings by bitmaps composed of a greeked text representation designed to indicate size and location of each such string at smaller scale~19020

**FIG. 190**